

1st Carlos A. Escobar
Global Research & Development
General Motors
Warren, MI, USA
carlos.1.escobar@gm.com

3rd Daniela Macias
Escuela de Ingenieria y Ciencias
Tecnologico de Monterrey
Monterrey, NL, Mexico
a01195037@itesm.mx

Abstract—Many real-world data sets contain missing values, therefore, learning with incomplete data sets is a common challenge faced by data scientists. Handling them in an intelligent way is important to develop robust data models, since there is no perfect approach to compensate for the missing values. Deleting the rows with empty cells is a commonly used approach, this *naive method* may lead to estimates with larger standard errors due to reduced sample size. On the other hand, imputing the missing records is a better approach, but it should be used with great caution, as it relies on often unrealistic specific assumptions which can potentially bias results. In this paper, a new greedy-like algorithm is proposed to maximize the number of records. The algorithm can be used to generate various maximized sub-sets by varying the number of columns (features) that can be used for learning. It salvages more records than the *naive method*, and it avoids the bias induced by imputation. The learning algorithms would be able to learn from real sub-sets without the bias induced by artificial data. Finally, the proposed algorithm is applied to a case study, the COVID-19 Open Research data set (CORD-19) that was prepared and posted by The White House and a coalition of leading research groups as a call to action to the world's artificial intelligence experts to answer high priority scientific questions. This data set contains missing records, therefore, resulting maximized sub-sets from this analysis can be further investigated by the research community.

I. INTRODUCTION

Deleting rows or columns with missing data is a widely used approach. However, as it will be illustrated, if this *naive*

	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2
--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	---

[illegible]

method is applied, oftentimes the sample size may be significantly reduced, leaving little information for the algorithm to learn the patterns. This method is not advised unless the proportion of eliminated records is very small ($<5\%$) [1].

Learning with incomplete data sets requires careful planning

and attention, since there is no perfect way to compensate for the missing values. Whereas removing the data will lead to estimates with larger standard errors due to reduced sample size, imputation can potentially bias results.

In this paper, a new greedy-like algorithm is proposed, it maximizes the number of total records that can be used to learn. As it will be demonstrated, the algorithm salvages more records than the *naive method*, and may help to reduce the bias induced by imputation, as the algorithms would be able to effectively learn from real data without the biased induced by artificially created data.

The rest of this paper is organized as follows. The proposed algorithm is presented in Section III. Followed by two real case studies in Section IV, one of them contributes in the pandemic, by formally analyzing the COVID-19 Open Research data set (CORD-19) that was posted by The White House as a call to action to the world's artificial intelligence experts to answers high priority scientific questions. Finally, conclusions and future research are presented in Section V.

II. STATE OF THE ART

In the current state of the art, handling missing values has been tested thoroughly. There is an existing concept, called missing data mechanism, where there is a probability for each data point to be missing [1]. That probability is governed by three categories, which are:

- Missing completely at random (MCAR) Where the probability of being missing is the same for all the data points in the set. This is due to unforeseen circumstances that are not related to a problem from the data.
- Missing at random (MAR) When the data is missing at random, the probability is caused by a group defined by the observed data. A circumstance that may cause missing data could vary from one condition to another. When those conditions are known, and the data is collected knowing the difference between those conditions, then the data will be MAR. This is a more realistic approach than MCAR, and may be the starting point for analysis.
- Not missing at random (NMAR) When there is a known cause to be missing data, which could be a situation or condition, then the data will be NMAR. The obtained data will need more in-depth analysis to handle and fix the origin of those missing values.

There is no distinction between MCAR, MAR or MNAR approaches. Practically, MCAR is unrealistic and the least plausible of the three, leaving MAR or MNAR mechanisms. For practical issues, modern missing data methods use the MAR assumption as the standard for analyzing the data.

Imputation is one of the most widely used methods for dealing with missing values; assuming the NMAR approach, but not knowing the data, could lead to use it in the wrong

way. As imputation mechanisms, obtaining mean and mode of the existing data, among other old techniques, and replacing it in the NaN values is proven to be ineffective, because imputation has to be adequate and follow the pattern of the NaN values [4]. Hot deck imputation has also shown inadequate results due to a high error probability caused by multiple missing values in the same row.

To avoid any possible bias induced by imputation methods, authors recommend -when possible- to apply the proposed algorithm to maximize the number of records used for learning. Thus, the learning algorithm will learn only real patterns. However, imputation methods should be evaluated and selected as part of the modeling solution to ensure estimates are readily available when needed, to keep the predictive systems running.

III. INFORMATION OPTIMIZATION ALGORITHM

The proposed algorithm is based on a greedy-like selection, algorithms in this category make whatever choice seems the best at each iteration. The pool of options depends on the choices made so far, but not on the future ones. It iteratively makes one greedy choice after another, reducing the problem into smaller subproblems. A greedy algorithm is not subject to combinatorial explosion, as it never reconsiders its choices, therefore, there is no guarantee of finding an optimal solution. The basic idea of the proposed algorithm, is to maximize the number of total records that can be used to learn. This problem is solved iteratively by a two-step approach that considers the number of empty (NaN) cells. First, the column(s) with the minimum number of NaN is selected, if there are two or more columns with the same number of NaNs, the one with the maximum number of associated NaNs is selected. Since the rows of the NaNs in the selected column will be deleted, in this second step the algorithm is selecting the column that will minimize the deletion of useful information. The algorithm has three components, inputs, outputs and initialization. The pseudo-code is presented in Fig. 2:

• Inputs

- X , matrix with m rows (samples) and n columns (features).

• Outputs

- $Order$, vector with the final order of the columns in the solution.

- $RowsSalvagedbyCol (RSC)$, vector with the number of rows included in the solution by column. Either constant or decreasing.

- $PercentofRowsSalvagedbyCol(PRSC)$, vector with the percentage of $RowsSalvagedbyCol$ relative to m .

- $CumulativeRecordsSalvagedbyCol(CRSC)$, vector with the number of records (cells) by column.

- $PercentofCumulativeRecordsSalvagedbyCol(PCRSC)$, vector with the percentage of $CumulativeRecordsSalvagedbyCol$ relative to the matrix size ($m \times n$).

- $TotalRecordsSalvaged(TRS)$, number that describes the

total number of records salvaged by the solution (*sum of RowsSalvagedbyCol*).

-*PercentofTotalRecordsSalvaged(PTRS)*, *Total-RecordsSalvaged* expressed in percentage relative to the matrix size ($m \times n$).

-*PermutedMatrix*, matrix sorted based on *Order*.

- **Initialization** -Define nc as 1 to n vector.
- Define Remaining as 1 to n vector.
- Define Tracker as 1 to n vector.
- Define SelectedF as an empty vector of size n .
- Define RSC as an empty vector of size n .
- Define PRSC as an empty vector of size n .
- Define CRSC as an empty vector of size n .
- Define PCRSC as an empty vector of size n .
- Define TRS as an empty vector of size n .
- Define PTRS as an empty vector of size n .

```

1. i = 1;
2. [m,n]=size(X);
3. for i = 1:n
4.   Define SumEmptyinRowbyCol as an empty vector of size n - i
5.   NumofEmptyinCol as an empty vector of size n - i
6.   SumDeleteshbyCol as an empty vector of size n - i
7.   for ii = 1:n
8.     SumEmptybyRow = 0;
9.     EmptyinCol = find(isnan(X(:,ii)));
10.    for iii = 1: numel(EmptyinCol);
11.      EmptybyRow = sum(isnan(X(EmptyinCol(iii),:)));
12.      SumEmptybyRow = SumEmptybyRow + EmptybyRow;
13.    end
14.    SumEmptyinRowbyCol(ii - i + 1) = SumEmptybyRow;
15.    NumofEmptyinCol(ii - i + 1) = numel(EmptyinCol);
16.    SumDeleteshbyCol(ii - i + 1) = NumofEmptyinCol(ii - i + 1) * n - SumEmptyinRowbyCol(ii - i + 1);
17.  end
18.  PreSelCol = find(min(NumofEmptyinCol) == NumofEmptyinCol);
19.  if numel(PreSelCol) > 1
20.    Selected = PreSelCol(find(max(SumEmptyinRowbyCol(PreSelCol)) == SumEmptyinRowbyCol(PreSelCol)));
21.  else
22.    Selected = PreSelCol;
23.  end
24.  SelectedF(i) = Remaining(Selected);
25.  Remaining = setdiff(nc,SelectedF);
26.  UpdatedOrder = [SelectedF(1:1),Remaining];
27.  [number,index] = ismember(UpdatedOrder,Tracker);
28.  X = X(index,index);
29.  Tracker = UpdatedOrder;
30.  Row2Del = find(isnan(X(:,1)));
31.  X(Row2Del,:) = [];
32.  [nswip,nskip] = size(X);
33.  Fills = NaN(m - nswip,1);
34.  Xwip(:,1) = [X(:,1);Fills];
35.  RowsSalvagedbyCol(i) = nswip;
36.  PercentofRowsSalvagedbyCol(i) = RowsSalvagedbyCol(i)/m;
37.  CumulativeRecordsSalvagedbyCol(i) = i + RowsSalvagedbyCol(i);
38.  PercentofCumulativeRecordsSalvagedbyCol(i) = CumulativeRecordsSalvagedbyCol(i)/(m * n);
39. end
40. TotalRecordsSalvaged = sum(RowsSalvagedbyCol);
41. PercentofTotalRecordsSalvaged = TotalRecordsSalvaged/(m * n);
42. Order = UpdatedOrder;
43. PermutedMatrix = Xwip;

```

Fig. 2. Records optimizer pseudo-code.

Lines 1-2 define initial values used by the algorithm. In lines 3-39, n number of iterations are performed to develop the solution, where n is the number of columns. In lines 4-6, the vectors used in this process are defined, since the number of remaining columns decreases by one after each iteration, their size also decreases by one. In lines 7-17, the information required to select the next column is generated. In lines 8-9, the variable that stores the sum of empties by row is reset and the positions of the NaNs by column are identified. In lines 10-13, these positions, are then used to determine the associated NaNs by row (line 11). The sum of these values is stored in line 12. The vectors defined in lines 4-6, are populated in lines 14-16. In line 18, the column with the minimum number of NaNs is selected. If there are two or more columns with the same number of NaNs, the column with the maximum number of associated NaNs by row is selected, this can be observed in lines 18-23. Line 24 populates the vector with

the solution. Line 25 keeps track of the remaining columns. Since the selected columns are moved to the left, their order needs to be updated, stated in line 26. In lines 27-28, the original matrix X is rearranged, based on the updated order. Line 29 helps to keep track of the original column number in each iteration. In lines 30-31, the rows of the empty cells in the selected column are deleted. In line 32, the size of the matrix is recorded. In lines 33-34, the *PermutedMatrix* is generated; the empty records of each preprocessed column are filled with NaN. In lines 35-43 statistics are reported and, at the end of the iterations, they store the information of the solution. Where *Order* contains the order of the matrix and *RowsSalvagedbyCol* contains the number of associated rows, this number exhibits a constant or a decreasing pattern. Finally, *PermutedMatrix* shows the overall solution.

A. Solution Process Overview

A virtual data set is created to show the solution step by step. The data set contains 11 columns and 10 rows with empty cells in the diagonal of columns 1-10, Table I. In this case, if rows with empty cells are eliminated, all the data set is deleted, if columns with empty cells are deleted only one column is salvaged (11), leaving very little information for the algorithm to learn from. Besides the data set, the table has two more rows at the bottom: (1) *Empty by column (EBC)*, which describes the number of empty cells in each column, and (2) *Associated empties (AE)*, which describes the sum of the row's associated empty cells to each of the empty cells in a column. Moreover, another column is added at the end which describes the number of empty cells by row, this information is used to determine the values of *AE*. A color convention is also included to keep track of the decisions and actions of the algorithm. The feasible options (i.e., minimum number of *EBC*) at each step are highlighted in gray, if there is a tie, the *AE* values (maximum) are used as tiebreaker to maximize the information extraction. If still, there are two or more columns with the same numbers, the smallest index is arbitrarily selected. The green color keeps track of the selected columns. Finally, based on the empty cells of the selected column, the yellow color is used to highlight which rows will be deleted. The full solution process is graphically described in Table II, the solutions are summarized in Table III and Table IV shows the *PermutedMatrix*. The solutions are interpreted as follows: in iteration one, column 11 is selected, which includes include 10 rows, the updated order column keeps track of selected columns (in bold) in each iteration, as well as remaining columns.

In iteration 1 (Table I), column 11 is selected since the *EBC* value is the smallest one (0), which means all the records in that column are available. In iteration 2 (upper left, Table II), there is a tie between columns eight and five, as both have a *EBC* = 1, however, column eight has three associated empty cells (*AE* = 3) vs two (*AE* = 2) of column five. In this situation, it is better to remove row eight (in yellow) to salvage more records. The remaining steps follow the same logic. Figure 3 shows how the number of records decreases (rows

TABLE I
VIRTUAL DATA SET.

		Features										Empty by row (EBR)	
Index		1	2	3	4	5	6	7	8	9	10		11
Samples		NaN	1	1	1	1	NaN	1	1	1	NaN	1	3
	2	NaN	2	2	2	2	2	2	2	2	2	2	1
	3	3	NaN	3	3	3	3	3	3	NaN	3	3	2
	NaN	4	4	NaN	4	4	NaN	4	4	4	4	4	3
	5	NaN	5	5	NaN	5	5	5	5	5	NaN	5	3
	6	6	NaN	6	6	NaN	6	6	6	6	NaN	6	3
	7	7	NaN	NaN	7	7	NaN	7	7	NaN	7	7	4
	NaN	8	NaN	NaN	8	8	8	NaN	8	8	8	8	4
	9	9	9	9	9	NaN	9	9	NaN	9	9	9	2
	10	10	10	10	10	10	10	10	10	10	NaN	10	1
Empty by column (EBC)		3	2	4	3	1	3	2	1	2	5	0	26
Associated empties (AE)		7	2	9	8	2	5	5	3	2	9	0	

TABLE II
SOLUTION PROCESS (ITERATIONS FROM LEFT TO RIGHT).

	11	1	2	3	4	5	6	7	8	9	10	EBR
	1	NaN	1	1	1	1	NaN	1	1	1	NaN	3
	2	2	NaN	2	2	2	2	2	2	2	2	1
	3	3	3	NaN	3	3	3	3	3	NaN	3	2
	4	NaN	4	4	NaN	4	4	NaN	4	4	4	3
	5	5	NaN	5	5	NaN	5	5	5	NaN	3	3
	6	6	6	NaN	6	6	NaN	6	6	6	NaN	3
	7	7	7	NaN	NaN	7	7	NaN	7	7	NaN	4
	8	NaN	8	NaN	NaN	8	8	8	NaN	8	8	4
	9	9	9	9	9	9	NaN	9	9	NaN	9	2
	10	10	10	10	10	10	10	10	10	10	NaN	1
EBC	0	3	2	4	3	1	3	2	1	2	5	26
AE	0	7	2	9	8	2	5	5	3	2	9	

	11	8	5	1	2	3	4	6	7	9	10	EBR
	1	1	1	NaN	1	1	1	NaN	1	1	NaN	3
	2	2	2	2	NaN	2	2	2	2	2	2	1
	3	3	3	3	3	NaN	3	3	3	NaN	3	2
	4	4	4	NaN	4	4	NaN	4	4	4	4	3
	6	6	6	6	6	NaN	6	NaN	6	6	NaN	3
	7	7	7	7	7	NaN	7	NaN	7	7	NaN	4
	9	9	9	9	9	9	9	NaN	9	NaN	9	2
	10	10	10	10	10	10	10	10	10	10	NaN	1
EBC	0	0	0	2	1	3	2	3	2	2	4	19
AE	0	0	0	4	0	6	5	5	2	7		

	11	8	5	2	4	1	3	6	7	9	10	EBR
	1	1	1	1	1	NaN	1	1	1	1	NaN	3
	3	3	3	3	3	3	NaN	3	3	NaN	3	2
	6	6	6	6	6	6	NaN	6	6	6	NaN	3
	9	9	9	9	9	9	NaN	9	9	NaN	9	2
	10	10	10	10	10	10	10	10	10	10	NaN	1
EBC	0	0	0	0	0	1	2	3	0	2	3	11
AE	0	0	0	0	0	2	3	5	0	2	4	

	11	8	5	2	4	7	1	3	6	9	10	EBR
	3	3	3	3	3	3	3	NaN	3	NaN	3	2
	6	6	6	6	6	6	6	NaN	6	NaN	6	3
	9	9	9	9	9	9	9	NaN	9	NaN	9	2
	10	10	10	10	10	10	10	10	10	10	NaN	1
EBC	0	0	0	0	0	0	0	2	2	2	2	8
AE	0	0	0	0	0	0	0	3	3	2	2	

	11	8	5	2	4	7	1	3	6	9	10	EBR
	10	10	10	10	10	10	10	10	10	10	NaN	1
EBC	0	0	0	0	0	0	0	0	0	0	1	1
AE	0	0	0	0	0	0	0	0	0	0	0	

	11	8	5	2	4	7	1	3	6	9	10	EBR	
	1	1	NaN	1	1	1	1	NaN	1	1	NaN	3	
	2	2	2	NaN	2	2	2	2	2	2	2	1	
	3	3	3	3	NaN	3	3	3	3	NaN	3	2	
	4	4	NaN	4	4	NaN	4	NaN	4	NaN	4	3	
	6	6	6	6	6	NaN	6	NaN	6	6	NaN	3	
	7	7	7	7	7	NaN	7	NaN	7	NaN	7	4	
	9	9	9	9	9	9	9	NaN	9	NaN	9	2	
	10	10	10	10	10	10	10	10	10	10	NaN	1	
EBC	0	0	0	2	2	3	2	1	3	2	2	5	22
AE	0	0	4	2	6	5	2	5	5	2	9		

	11	8	5	2	1	3	4	6	7	9	10	EBR	
	1	1	1	1	NaN	1	1	NaN	1	1	NaN	3	
	3	3	3	3	3	NaN	3	3	3	3	NaN	3	
	4	4	4	4	NaN	4	NaN	4	NaN	4	4	3	
	6	6	6	6	6	NaN	6	NaN	6	6	NaN	3	
	7	7	7	7	7	NaN	7	NaN	7	NaN	7	4	
	9	9	9	9	9	9	9	NaN	9	NaN	9	2	
	10	10	10	10	10	10	10	10	10	10	NaN	1	
EBC	0	0	0	0	2	3	2	3	2	2	4	18	
AE	0	0	0	0	4	6	5	3	5	2	7		

	11	8	5	2	4	7	1	3	6	9	10	EBR	
	1	1	1	1	1	1	1	NaN	1	NaN	1	3	
	3	3	3	3	3	3	3	3	NaN	3	NaN	3	
	6	6	6	6	6	6	6	6	NaN	6	NaN	3	
	9	9	9	9	9	9	9	9	NaN	9	NaN	2	
	10	10	10	10	10	10	10	10	10	10	NaN	1	
EBC	0	0	0	0	0	0	0	1	2	3	2	3	11
AE	0	0	0	0	0	0	0	2	3	3	2	4	

	11	8	5	2	4	7	1	3	6	9	10	EBR	
	9	9	9	9	9	9	9	9	NaN	NaN	9	2	
	10	10	10	10	10	10	10	10	10	10	NaN	1	
EBC	0	0	0	0	0	0	0	0	0	1	1	3	
AE	0	0	0	0	0	0	0	0	0	0	0		

	11	8	5	2	4	7	1	3	6	9	10	EBR	
	10	10	10	10	10	10	10	10	10	10	NaN	1	
EBC	0	0	0	0	0	0	0	0	0	0	1	1	
AE	0	0	0	0	0	0	0	0	0	0	0		

Solution													
	11	8	5	2	4	7	1	3	6	9	10		
	10	10	10	10	10	10	10	10	10	10	NaN	1	
EBC	0	0	0	0	0	0	0	0	0	0	0	1	
AE	0	0	0	0	0	0	0	0	0	0	0		

salvaged) by including more columns, whereas the number of cumulative records increases and eventually decline. In this case, if all columns are included, no rows are salvaged.

This algorithm addresses the tradeoff between the number of columns (features) and rows (samples). In real life problems, this is a common situation, as an example, medical files of patients tend to have a lot of missing records, or in manufacturing, as described earlier, sensors may randomly break down or may be systematically turned off. Whereas this algorithm helps to maximize the information extraction, the main question remains: which sub-data set should be used to

learn? In this context, this algorithm cannot straight forward answer this question, instead, it is a guide on how to cleverly define sub-data sets to explore. For example, in this data set, two useful insights can be derived, per Table III, the sub-data set generated in iteration six is a better solution than the sub-data set generated in iteration five, as both have the same number of rows but the former has more features (6 vs 5) or cumulative records (30 vs 25). Finally, if all features are included, there are no samples left to learn from. In real life problems, this situation refers to the columns on the extreme right, that oftentimes have very little information, making the

TABLE III
SOLUTIONS.

Iteration	Selected column	Rows salvaged	Columns included	Cumulative cells	Updated order
1	11	10	1	10	11,1,2,3,4,5,6,7,8,9,10
2	8	9	2	18	11,8,1,2,3,4,5,6,7,9,10
3	5	8	3	24	11,8,5,1,2,3,4,6,7,9,10
4	2	7	4	28	11,8,5,2,1,3,4,6,7,9,10
5	4	5	5	25	11,8,5,2,4,1,3,6,7,9,10
6	7	5	6	30	11,8,5,2,4,7,1,3,6,9,10
7	1	4	7	28	11,8,5,2,4,7,1,3,6,9,10
8	3	2	8	16	11,8,5,2,4,7,1,3,6,9,10
9	6	1	9	9	11,8,5,2,4,7,1,3,6,9,10
10	9	1	10	10	11,8,5,2,4,7,1,3,6,9,10
11	10	0	11	0	11,8,5,2,4,7,1,3,6,9,10

TABLE IV
PermutedMatrix, OVERALL SOLUTION.

Features											
11	8	5	2	4	7	1	3	6	9	10	
1	2	3	4	5	6	7	8	9	10	NaN	NaN
2	3	4	5	6	7	8	9	10	NaN	NaN	NaN
3	4	5	6	7	8	9	10	NaN	NaN	NaN	NaN
4	5	6	7	8	9	10	NaN	NaN	NaN	NaN	NaN
5	6	7	8	9	10	NaN	NaN	NaN	NaN	NaN	NaN
6	7	8	9	10	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	8	9	10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	9	10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

data set ineffective, as illustrated in the following section.

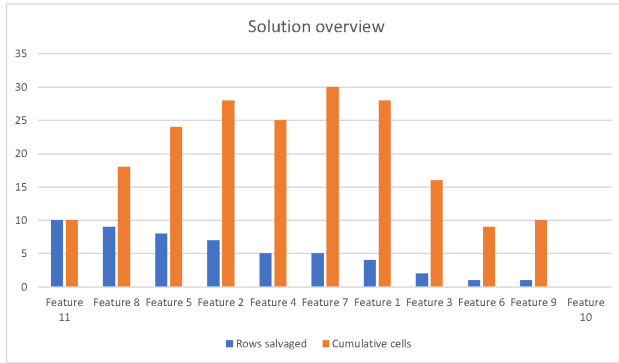


Fig. 3. Salvaged rows and cumulative cells by columns.

IV. CASE STUDY

In this section, two public data sets are analyzed, they are particular relevant because they were not preprocessed before being posted, therefore they have a lot of missing records: (1) COVID-19 Open Research Data Set [5], this data set calls to action to the world's artificial intelligence experts to develop text and data mining tools that can help the medical community develop answers to high priority scientific questions, in this context, by analyzing this data set, we support the ongoing COVID-19 response efforts worldwide.

A. COVID-19 Open Research Data Set

This data set contains 111 features and 5644 samples, most of its cells are empty, as shown in Fig. 4. If rows with empty cells are eliminated, no rows would be salvaged, whereas if columns with empty cell are eliminated, only six columns would be salvaged, leaving the potential contribution of some of the 105 remaining features out of the learning part. The proposed algorithm is applied, Table V displays the set of solutions, in which column one describes the number of features, column two the feature index, and column three describes the number of salvaged rows, refer to Table VI in Appendix A to match the indexes with the actual names. Based on the set of solutions, a couple of relevant sub-data sets are exhibited, the first sub-data set would contain six features with 5644 records, the second data set would contain 23 features with 1352 records, then the third data set would contain 37 features with 362 records, if necessary, a few more sub-data sets can be created, but they would contain very little information, this information is graphically displayed in Fig. 5.

V. CONCLUSIONS AND FUTURE WORK

Learning with incomplete data sets is a common challenge faced by data scientists. Handling them in an intelligent way is important to develop robust data models. While deleting rows/columns or imputing missing values are common approaches, they may have serious implications such as losing most of the training data (as demonstrated in the real case studies) or biasing the estimates. In this context, the authors



Fig. 4. Screen-shot of COVID-19 data set, empty cells in black.

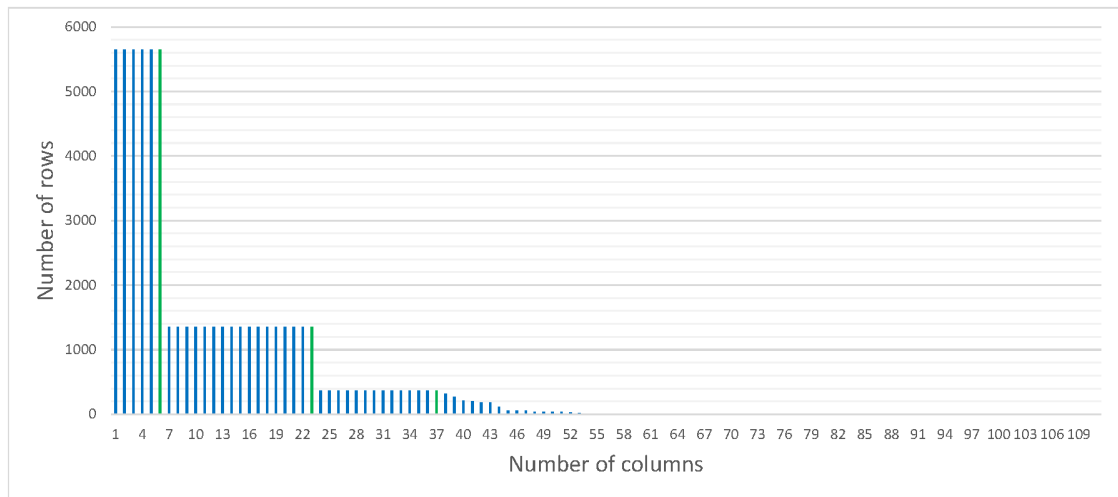


Fig. 5. Salvaged rows by number of columns, the green color describes good data sets.

advocate –when possible– to learn from real data and impute missing values on deployment.

In this paper, a new greedy-like algorithm is presented, the algorithm is aimed at maximizing the information (cells) of incomplete data sets. It develops different sub-sets that solve the trade-off between number of features and number of samples differently. The user can choose a sub-set with either more features and less number of samples or the other way around. Since the best sub-set cannot be determined in advance, it is recommended to apply the learning algorithm to select the best, like wrapper-type feature selection algorithms [6].

Moreover, this paper reports a small contribution in addressing the pandemic challenge from an analytical perspective, by formally analyzing the COVID-19 Open Research Data Set (CORD-19) that was prepared and posted by The White House and a coalition of leading research groups as a call to action to the world’s artificial intelligence experts to answer high priority scientific questions.

The proposed algorithm iteratively considers the number of empty cells in each remaining column to select the next one in a greedy fashion. This study can be enhanced by considering the predictive information associated to each columns too. This information can be obtained by using a filter-type feature selection algorithm [6]. Thus, columns are selected based on two criteria, the number of cells and their quality.

TABLE V
SOLUTIONS, THE GREEN COLOR DESCRIBES GOOD DATA SETS.

Num. of columns	Feature index	Num. of rows	Num. of columns	Feature index	Num. of rows
1	1	5644	57	59	6
2	2	5644	58	60	6
3	3	5644	59	61	6
4	4	5644	60	63	6
5	5	5644	61	64	6
6	6	5644	62	65	6
7	22	1354	63	46	1
8	23	1354	64	47	1
9	24	1354	65	72	1
10	25	1352	66	73	1
11	26	1352	67	74	1
12	27	1352	68	75	1
13	29	1352	69	76	1
14	30	1352	70	77	1
15	31	1352	71	79	1
16	32	1352	72	80	1
17	33	1352	73	81	1
18	34	1352	74	84	1
19	35	1352	75	85	1
20	36	1352	76	86	1
21	37	1352	77	87	1
22	38	1352	78	88	1
23	39	1352	79	89	1
24	7	366	80	28	0
25	8	366	81	55	0
26	9	366	82	56	0
27	11	366	83	57	0
28	12	366	84	62	0
29	13	366	85	66	0
30	14	366	86	67	0
31	15	366	87	68	0
32	16	366	88	69	0
33	17	366	89	70	0
34	18	366	90	71	0
35	20	366	91	78	0
36	19	365	92	82	0
37	10	362	93	83	0
38	42	320	94	90	0
39	40	272	95	94	0
40	43	212	96	95	0
41	41	201	97	97	0
42	45	179	98	98	0
43	44	178	99	99	0
44	21	114	100	100	0
45	51	52	101	101	0
46	52	52	102	102	0
47	53	52	103	103	0
48	48	41	104	104	0
49	49	41	105	105	0
50	50	36	106	106	0
51	54	36	107	107	0
52	92	27	108	108	0
53	91	19	109	109	0
54	96	11	110	110	0
55	93	10	111	111	0
56	58	6			

APPENDIX A

This appendix includes the table with the names of the COVID-19 features.

TABLE VI
NAMES OF COVID-19 FEATURES.

Index	Feature name	Index	Feature name
1	Patient ID	57	Magnesium
2	Patient age quantile	58	pCO ₂ (venous blood gas analysis)
3	SARS-Cov-2 exam result	59	Hb saturation (venous blood gas analysis)
4	Patient addmitted to regular ward (1=yes, 0=no)	60	Base excess (venous blood gas analysis)
5	Patient addmitted to semi-intensive unit (1=yes, 0=no)	61	pO ₂ (venous blood gas analysis)
6	Patient addmitted to intensive care unit (1=yes, 0=no)	62	Fio ₂ (venous blood gas analysis)
7	Hematocrit	63	Total CO ₂ (venous blood gas analysis)
8	Hemoglobin	64	pH (venous blood gas analysis)
9	Platelets	65	HCO ₃ (venous blood gas analysis)
10	Mean platelet volume	66	Rods #
11	Red blood Cells	67	Segmented
12	Lymphocytes	68	Promyelocytes
13	Mean corpuscular hemoglobin concentration (MCHC)	69	Metamyelocytes
14	Leukocytes	70	Myelocytes
15	Basophils	71	Myeloblasts
16	Mean corpuscular hemoglobin (MCH)	72	Urine - Esterase
17	Eosinophils	73	Urine - Aspect
18	Mean corpuscular volume (MCV)	74	Urine - pH
19	Monocytes	75	Urine - Hemoglobin
20	Red blood cell distribution width (RDW)	76	Urine - Bile pigments
21	Serum Glucose	77	Urine - Ketone Bodies
22	Respiratory Syncytial Virus	78	Urine - Nitrite
23	Influenza A	79	Urine - Density
24	Influenza B	80	Urine - Urobilinogen
25	Parainfluenza 1	81	Urine - Protein
26	CoronavirusNL63	82	Urine - Sugar
27	Rhinovirus/Enterovirus	83	Urine - Leukocytes
28	Mycoplasma pneumoniae	84	Urine - Crystals
29	Coronavirus HKU1	85	Urine - Red blood cells
30	Parainfluenza 3	86	Urine - Hyaline cylinders
31	Chlamydia pneumoniae	87	Urine - Granular cylinders
32	Adenovirus	88	Urine - Yeasts
33	Parainfluenza 4	89	Urine - Color
34	Coronavirus229E	90	Partial thromboplastin time (PTT)
35	CoronavirusOC43	91	Relationship (Patient/Normal)
36	Inf A H1N1 2009	92	International normalized ratio (INR)
37	Bordetella pertussis	93	Lactic Dehydrogenase
38	Metapneumovirus	94	Prothrombin time (PT), Activity
39	Parainfluenza 2	95	Vitamin B12
40	Neutrophils	96	Creatine phosphokinase (CPK)
41	Urea	97	Ferritin
42	Proteina C reactiva mg/dL	98	Arterial Lactic Acid
43	Creatinine	99	Lipase dosage
44	Potassium	100	D-Dimer
45	Sodium	101	Albumin
46	Influenza B, rapid test	102	Hb saturation (arterial blood gases)
47	Influenza A, rapid test	103	pCO ₂ (arterial blood gas analysis)
48	Alanine transaminase	104	Base excess (arterial blood gas analysis)
49	Aspartate transaminase	105	pH (arterial blood gas analysis)
50	Gamma-glutamyltransferase	106	Total CO ₂ (arterial blood gas analysis)
51	Total Bilirubin	107	HCO ₃ (arterial blood gas analysis)
52	Direct Bilirubin	108	pO ₂ (arterial blood gas analysis)
53	Indirect Bilirubin	109	Arterial Fio ₂
54	Alkaline phosphatase	110	Phosphor
55	Ionized calcium	111	ctO ₂ (arterial blood gas analysis)
56	Strepto A		

REFERENCES

- [1] J. C. Jakobsen, C. Gluud, J. Wetterslev, and P. Winkel, "When and how should multiple imputation be used for handling missing data in randomised clinical trials—a practical guide with flowcharts," *BMC medical research methodology*, vol. 17, no. 1, p. 162, 2017.
- [2] J. Barnard and X.-L. Meng, "Applications of multiple imputation in medical studies: from aids to nhanes," *Statistical methods in medical research*, vol. 8, no. 1, pp. 17–36, 1999.
- [3] M. M. Rahman and D. N. Davis, "Machine learning-based missing value imputation method for clinical datasets," in *IAENG transactions on engineering technologies*. Springer, 2013, pp. 245–257.

- [4] P. Royston, "Multiple imputation of missing values," *The Stata Journal*, vol. 4, no. 3, pp. 227–241, 2004.
- [5] G. U. C. f. S. This dataset was created by the Allen Institute for AI in partnership with the Chan Zuckerberg Initiative, I. Emerging Technology, Microsoft Research, i. c. w. T. W. H. O. o. S. the National Library of Medicine National Institutes of Health, and T. Policy., "Covid-19 open research dataset challenge (cord-19)," kaggle, Aug. 2020. [Online]. Available: <https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge>
- [6] G. Chandrashekar and F. Sahin, "A Survey on Feature Selection Methods," *Computers & Electrical Eng*, vol. 40, no. 1, pp. 16–28, 2014.