Dementia

November 30, 2020

Contents

1 Data Sources

1.1 Datasets Used

The datasets used are from the following dementia surveys conducted in Lebanon:

- Bekaa Questionnaire: This dataset is the most recent survey conducted last year in the Bekaa. This dataset involved 219 elderly.

- full validation data: This data consists of 281 rows. This is the very first survey that we implemented for the sake of testing the 1066 Dementia SPSS algorithm. This survey was conducted on a 1-1 ratio with clinically diagnosed demented patients.
- dementia data 502 baseline updated: This is the second survey implemented which involved 502 elderly people. The selection of elderly to participate in this survey was random.

We then merged this dataset into one dataset consisting of a total of 1024 elderly.

1.2 Dementia Output

In order to label each row (elderly) as positive (has dementia) or negative (does not have dementia), we run the 1066 Dementia SPSS algorithm on our dataset. The result was a total of 1024 patients, 203 of whom have dementia according to the SPSS algorithm.

2 Split Into numeric and textual

We have split the data into two chunks, one containing numeric features and another containing categorical features. The Bekaa and full validation questionnaires involved textual questions as well. However, these textual questions are not used by the dementia 1066 SPSS algorithm in determining whether the elderly has dementia or not, so we dropped these textual features from the analysis.

3 Numeric Features

Our data is now a mix of numerical and categorical features. For the numeric features, we have obtained the following information:

- **data type:** Type of the numeric feature. Could possible be:
  1. **numeric** meaning this feature has continuous range of values
  2. **categorical** meaning this feature has only a finite set of values, each resembling a specific category
  3. **ordinal** meaning, this feature's values are categorical, but they can be compared between each other. Example would be: a job, were a specific job might be values more than the other. Example: manager vs assistant

  the specification of the **data type** of each feature was done manually by looking at the range of values in each, and also by looking at the choices workbook in the copy of dementia Excel questionnaire

- **Description:** Description of each feature
- **cat_options:** Short for "categorical options". *In case the feature is categorical*, what are the possible categories.
- **val_range:** Short for "values range". *In case the feature is numerical*, what are the range of values this feature takes
- **min:** *In case the feature is numeric*, what is the minimum value it has
- **max:** *In case the feature is numeric*, what is the maximum value it has
- **distribution:** link to the distribution plot of each feature on bitbucket. The distribution plot helps in detecting erroneous data
- **perc_missing:** Short for "percentage of missing values". This helps us ggregate features with high percentage of missing. In case the percentage of missing is very high, Imputation methods won't aid much
- **perc_erroneous:** Short for "percentage of erroneous values". Detecting erroneous values was done manually for each feature. Anomalies cover uni-intentional or intentional erroneous values. Example would be having "age" feature being 1966 instead of the actual age
- **erroneous:** What were the erroneous values, if any
- **cut_off:** The cut off used to decide whether a feature's value is erroneous or not. The cut off was discovered manually by us for each feature.
- **color code:** a color code was applied to each feature to determine whether it belongs to the informant or not.

3.1 Informant and High Percentage of Missing

Most of the features associated with very high percentage of missing were directed to informants and related to the living conditions of the family, the house, income, cor-residents of the elderly, etc... which are not used by the SPSS algorithm to label the patients as demented or not (since they are not about the patients) For this reason, we dropped the questions that were about the household, informants, and co-residents

* informant is usually the caregiver of the interviewed elderly person

3.2 Nested Questions and High percentage of missing

Since our data is survey in nature, some features are actually questions are related to the other. If a certain question is asked to the elderly, and only based upon the elderly's answer, the interviewer might ask the elderly another question or not. Let us give an example:

1. the interviewer asks if there has ever been a period when the elderly smoked cigarettes, cigars, a pipe, or water pipe nearly every day?
2. **Only if the answer to the question above was a Yes**
3. the interviewer asks the following questions:
    (a) What did the elderly smoke?
    (b) How old were you when you started using tobacco regularly? (if the above is true)
    (c) Do you still use tobacco regularly?

We also realized that many features have high percentage of missing because a parent question's answer did not cause a jump to that feature question.

3.3 Treating Features with High Percentage of Missing

We divided the features as follows:

1. **Informant** A feature belongs to this category if the question is asked to the informant
2. From Informants we have:
    (a) **parent** Questions that caused jumps to other questions
    (b) **child** child of parent question
3. **Non-Informant** We also have:
    (a) **parent**
    (b) **child**

We followed the following steps in order to eliminate features with high percentage of missing:

1. remove all informants and their children questions (so we removed all informants, and if one of them happens to be a parent question, we removed all its children questions)
2. It is important to note that all parent questions informants have their children questions also informant
3. Then we are left with **Legal features**
4. From legal features, remove all children

After doing the steps above, the number of numerical features decreased from 542 to 257, all of which have 0% missing (with the exception of 1 feature whcih has 50% missing)

3.4 Detecting Erroneous Values Inside Features

In our dataset, the categorical and ordinal values are encoded with numbers ranging from 1 to the length of the categories. therefore, since our categorical, ordinal, and numerical values are all numbers, we set cut-offs to determine erroneous values. For some of the features, the erroneous values are straight forward to detect, based on the description for the question. These are the following:

- age: some people enter the year, others enter the age. for this, if the value of both the numb as is or 2020 - the numb is more than 100, this means that the person is more than 100 years old which renders the age erroneous. For age, the cut-off was 100
- helphour: number of hours per week the elderly needed help, therefore bounded by max numb of hours in a week
- learn questions: the patient is supposed to repeat 3 words, bounded by the max numb of words: 3

For the rest of the categorical and ordinal values, we get the maximum number encoding for the question options from the options excel sheet based on which the interviewers selected these options. The cut-off is therefore the maximum number encoding a category. Were therefore label any value that is greater than the cut-off to be erroneous.

3.5 Working with Legal Features

As mentioned in section 3.3, the remaining ***"Legal"*** Features are the 257 out of originally 542. We will be using these 257 ***"Legal"*** features as input for data pre-processing and cleaning techniques.

3.6 Filtering Legal Features with Erroneous Values - Erroneous Code-Book

In section 3, we talked about how some of the 542 numeric features have erroneous values. However, as we will be working with only "Legal" features, its important to filter out which subset of the features with erroneous values are actually "Legal". Thats's why, we did the following:

1. From all the features containing erroneous, filter out the ones that are Legal
2. From the Legal ones:
   (a) get the feature name
   (b) get the feature's description
   (c) get the feature's percentage of erroneous values

(d) get the feature's actual erroneous values

(e) Display the feature's cut-off on values which decides which values are erroneous (crossing the cut-off) and which are non-erroneous (not crossing the cut-off)

3. When done from 2. above, sort the features by decreasing order of percentage of erroneous values

3.7 Detecting Outliers

We want to investifgate the existsence of outliers in the data. We have 3 different data types:

1. **Numeric**: very few columns
2. **Ordinal**: categorical values that obey a certain order of importance
3. **Categorical**

The outlier detection methods are known for numeric values, but it is **ordianl** and **categorical** values that raise the question of how are we going to detect outliers with such types of data ?
We will assume that ordinal values are numeric. We will prove the legit-ability of our assumption through an example:
Assume for instance, that we have a column that shows the socio economic status ("low income","middle income","high income"), education level ("high school","BS","MS","PhD"), income level ("less than 50K", "50K-100K", "over 100K"), satisfaction rating ("extremely dislike", "dislike", "neutral", "like", "extremely like"). These are not categorical but rather ordinal, and if we give, for example, the income level the following labels:

- less than 50K: 0
- 50K-100K: 1
- over 100K: 2

Then it is definitely the case that $2 > 1 > 0$. And if we have an individual who earns, > 1000 K, we give this the label 3, and these individuals are definitely rare and can be considered as outliers

3.8 Outlier Detection and Scaling

We try different scaling methods before we detect outliers, and we apply the outlier detection only for columns *that are: numeric and/or ordinal*, and we extracted the percentage of outliers for each of the legal columns as well as the outlier values themselves (values considered outliers).
We realize that the outlier values, for the **ordinal** columns, happen to be the values *8 and 9* which is normal because the values are either 0,1,2 or 8,9.

3.9 Feature Cross

Our data mostly consists of categorical/ordinal columns, and we have to find a way to solve these non-linearities such that it fairs better for the model's understanding of the features when we begin with the modelling phase. We generate the list of all possible feature crosses.
Feature crossing, will come after we 'one-hot encode' our data.

4 Data Transforms (Hiyam)

4.1 Scaling Numeric Data (chapter 17)

- Normalization (Min-Max Scaling)
- Standardization

4.2 Scaling Data with Outliers (chapter 18)

Many machine learning algorithms perform better when **numerical** input variables are scaled. Standardization is a popular scaling technique that substracts the mean from the values and divide by the stadard deviation, transforming the probability distribution from for an input variable to a Standard Gaussian (zero mean and unit variance).
Problem: Standardization can become skewed or biased when the input variable contains outliers.

Robust Scaling

When we are scaling the data, and if we have very large values relative to the other input variables, these large values can dominate or skew some machine learning algorithms. **The result is that the algorithms pay most of their attention on the large values and ignore the variables with smaller values**.
Outliers are values at the edge of the distribution that may have a low probability of occurence, yet are overrepresented for some reason. **Outliers can skew a probability distribution and make data scaling using standardization difficult as the calculated mean and standard deviation will be skewed by the presence of outliers.**
Approach: When standardizing input variables containing outliers, we can ignore outliers from the calculation of the mean and standard deviation, and use the calculated values to scale the data
This is called robust standardization. This can be achieved by calculating the median (50th percentile) and the 25th and 75th percentile. The values of each variable can then have their median subtracted and are divided by the

inter quartile range (IQR) which is the difference between the 25th and 75th percentile.

$$value = \frac{value - median}{p_{75} - p_{25}} \tag{1}$$

The resulting value has a **zero mean and median and a standard deviation of 1**. Although not skewed by outliers and the outliers are still present with the same relative relationships to other values.

4.3 How to Encode Categorical Data (chapter 19)

Machine learning models require all input and output variables to be numeric. This means that if the data contains categorical data, we must encode it to numbers before we fit and evaluate our models.

Ordinal Encoding

Each unique category is assigned an integer value. Example: *red* is 1, *green* is 2, *blue* is 3.
For categorical variables, it imposes and **ordinal relationship** when no such relationship exists. This may cause problems and **one hot encoding** may be used instead.

One Hot Encoding

For categorical variable where no ordinal relationship exists, the ordinal encoding is not enough and may be misleading.
One Hot Encoding works by creating binary variables for each category. For Example: *red* will be [1, 0, 0], *blue* will be [0, 1, 0], and *green* will be [0, 0, 1].

Dummy Variable Encoding

The one hot encoding includes a binary representation for each category. This might cause redundancy.
if we know that [1, 0, 0] represents *blue*, and [0, 1, 0] represents *green*, we don't need another binary variable to represent *red*, instead we could use 0 values along, e.g. [0, 0]. This is called dummy variable encoding and always represents C categories with $C-1$ binary variables.
Other being less redundant, it is required for some models.

4.4 How to Make Distributions Look More Gaussian (chapter 20)

Several Machine Learning Algorithms assume the numerical values have a Gaussian distribution. Our data may not have a Gaussian distribution and it might have a Gaussian-like distribution.