DATA PRE-PROCESSING IN MATLAB

Author: Abisoye Akinloye

Connet with me: https://linktr.ee/abisoye.akinloye

Source codes: Github link

Table of Contents

Data Processing

It is an important steps in building Machine Learning model. If not done, the ML model wouldn't work effectively.

Why data preprocessing?

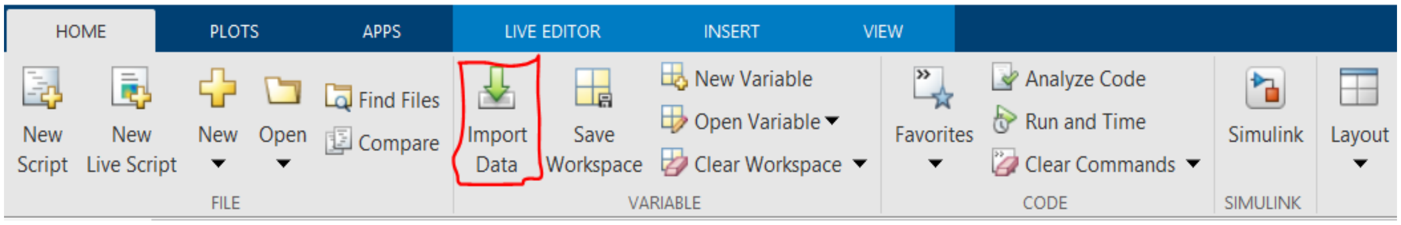
- Data Cleaning
- Data Transformation
- Data Reduction

The first step in data preprocessing is to import the data. Data can be imported in MATLAB as follows:

Importing Data

There are two ways of importing data in MATLAB:

- From the **Home** tab `Import Data` option



- Reading the data as table, csv, or delimiter method.

```
% read csv
data = csvread("data.csv")

% delimiter method
data = dlmread("data.txt",",")

% read table (recommended for ML project)
data = readtable(filename)
```

Note:

- If the imported data comprise of missing numerial value, it is replaced with `NaN (Not a number)`
- If the imported data comprise of missing textual value (character), it is replaced with `'' (Empty character)`
- If the imported data comprise of missing categorical value (string), it is replaced with `<undefined>`
- Machine learning models require **`table`** data type

```
% import data
data = readtable("..\data\data_1.csv","VariableNamingRule","preserve");

% Variable naming rule set as preserved to keep the column name the same
% when it consists white space.

% check the first 5 rows of the data
data(1:5,:)
```

```
ans = 5×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	NaN	'Liked'

The data above show the opinion (liked or not liked) of different buyers of a goods based on their location, age, annual salary.

- The opinion is the **`dependent`** variable

- The other features are the **`independent`** variables.

Handling Missing Numerical Data

There are methods of handling missing data. They are:

- Deleting rows or colums with missing value.
- Replacing the missing data with the `mean/median` of the column for numerical data.
- Replacing the missing categorical data with `mode.`
- To check if there is a missing value, use the built-in function.

```
% check if there is missing data
ismissing(data)

% delete mising rows
rmmissing(data)

% delete missing column
rmmissing(data,2)
```

Deleting rows or columns with missing values

```
% delete rows of missing value
no_missing_row_data = rmmissing(data)
```

```
no_missing_row_data = 8×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'US'	33	58000	'Liked'
6	'US'	40	79000	'Liked'
7	'Africa'	55	83000	'not liked'
8	'US'	35	67000	'Liked'

```
% delete columns with missing values
no_missing_col_data = rmmissing(data,2);

% get first 5 rows of the data
no_missing_col_data(1:5,:)
```

```
ans = 5×2 table
```

	Location	Opinion
1	'US'	'not liked'

	Location	Opinion
2	'Asia'	'not liked'
3	'Africa'	'not liked'
4	'Asia'	'not liked'
5	'Africa'	'Liked'

Note:

- By default using `rmmissing()` will remove **rows** with missing value
- Add `argument/dimension` **2** to remove column with missing values.
- Deleting columns or rows can be more dangerous to the data, however, it is not recommended.
- If the data is very large and the missing value is not very significant, you could delete the rows.
- Deleting columns should be generally **avoided** and should only be considered when numbers of features are many.
- For example, in text categorization and email classification problems.

Deleting Rows or Columns with Relative percentage of missing value

- Add the **`MinNumMissing`** argument to the `rmmissing()`

```
% import and show the data
data2 = readtable("..\data\data_2.csv","VariableNamingRule","preserve");
data2(1:5,:)
```

```
ans = 5×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	NaN	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	NaN	NaN	'Liked'

```
% 2 -> least number of missing value in a row
rel_data = rmmissing(data2,'MinNumMissing',2)
```

```
rel_data = 9×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	NaN	'not liked'
4	'Asia'	35	61000	'not liked'
5	'US'	33	58000	'Liked'
6	'Asia'	NaN	52000	'not liked'

	Location	Age	Annual Salary	Opinion
7	'US'	40	79000	'Liked'
8	'Africa'	55	83000	'not liked'
9	'US'	35	67000	'Liked'

```
% 2 -> least number of missing value in a column to be deleted
% N.B: It is not recommended as the necessary features have been deleted
rel_data_del = rmmissing(data2,2,"MinNumMissing",2)
```

```
rel_data_del = 10×2 table
```

	Location	Opinion
1	'US'	'not liked'
2	'Asia'	'not liked'
3	'Africa'	'not liked'
4	'Asia'	'not liked'
5	'Africa'	'Liked'
6	'US'	'Liked'
7	'Asia'	'not liked'
8	'US'	'Liked'
9	'Africa'	'not liked'
10	'US'	'Liked'

Replacing missing value with Mean

- Use `mean()` and add `omitnan` argument to compute the mean.
- Use `fillmissing()` to fill the missing value
- The `fillmissing` methods are: `constant`, `next`, `previous`, `nearest`, and so on.

```
% calculate mean omitting NaN
age_mean = mean(data.Age,'omitnan');

% fill missing age with a constant value -> mean
no_missing_age = fillmissing(data.Age,"constant",age_mean);

% copy the data (not necessary)
copy_data = data;

% replace the age column with age with no missing value
copy_data.Age = no_missing_age
```

```
copy_data = 10×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'

	Location	Age	Annual Salary	Opinion
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	NaN	'Liked'
6	'US'	33	58000	'Liked'
7	'Asia'	37.444	52000	'not liked'
8	'US'	40	79000	'Liked'
9	'Africa'	55	83000	'not liked'
10	'US'	35	67000	'Liked'

```
% replace missing value in salary column
salary_mean = mean(data.("Annual Salary"), 'omitnan');
no_missing_salary = fillmissing(data.("Annual Salary"),"constant",salary_mean);

format shortG
copy_data.("Annual Salary") = no_missing_salary
```

```
copy_data = 10×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	63778	'Liked'
6	'US'	33	58000	'Liked'
7	'Asia'	37.444	52000	'not liked'
8	'US'	40	79000	'Liked'
9	'Africa'	55	83000	'not liked'
10	'US'	35	67000	'Liked'

Handling missing textual data

Textual data cannot be replaced with the `mean`. You cannot take the mean of non-numerical value. Hence, **find and replace by most frequently** occuring value.

- Covert to categorical data type `categorical()`
- Find the most occuring value `mode()`
- Fill with the most occuring value.

```
% convert to categorical data
categorical(data)
```

```
% most occuring
m = mode(data.VariableNames)

% fill missing
fillmissing(data.VariableNames, 'constant' cellstr(m))
% You need to convert the cell to string inorder to fill it in the categorical data
```

```
% import the data
data3 = readtable("..\data\data_3.csv","VariableNamingRule","preserve");
data3(1:5,:)
```

```
ans = 5×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	''
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	NaN	'Liked'

```
% first convert to categorical data type
data3.Opinion = categorical(data3.Opinion);

% find the mode
freq_opinion = mode(data3.Opinion);

% fill missing value with most frequent
no_missing_opinion = fillmissing(data3.Opinion,"constant",cellstr(freq_opinion));
data3.Opinion = no_missing_opinion;
data3(1:5,:)
```

```
ans = 5×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	not liked
2	'Asia'	25	48000	not liked
3	'Africa'	30	54000	not liked
4	'Asia'	35	61000	not liked
5	'Africa'	44	NaN	Liked

Features scaling

```
% import the data
data4 = readtable("..\data\data_4.csv","VariableNamingRule",'preserve');
data4(1:5,:)
```

```
ans = 5×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	63777	'Liked'

Considering the features (Age and Annual Salary), it is obvious that both are not on the same scale/range. The value is of annual salary is much greater than age. This will however lead to issues in the Machine Learning model. When those features are fed into ML model, it assumes that age value is negligible; thereby making prediction based on annual salary only.

Hence, you can transform the features into the same scale for instance `-1 or +1` whereby the minimum value can rep -1 while the maximum represents +1.

There are different types of features scaling, but the most often used are:

- **Standardization**
- **Normalization**

Standardization

It transforms features from the range of `-1 to +1`

$$\mathbf{x_{transformed}} = \frac{\mathbf{x - mean(x)}}{\mathbf{standard\ deviation(x)}}$$

```
% Create a M-file named "standardize.m" and enter the following lines of code

% Function for standardized feature scaling
function x_transformed = standardize(feature)
    if sum(ismissing(feature)) == 0
        % mean of the features
        M_feature = mean(feature);
        % standard deviation of features
        Std_feature = std(feature);
        % standardize
        x_transformed = (feature - M_feature)/Std_feature;
    else
        % mean of the features
        M_feature = mean(feature,'omitnan');
        % standard deviation of features
        Std_feature = std(feature,'omitnan');
        % standardize
        x_transformed = (feature - M_feature)/Std_feature;
    end
end
```

Normalization

It transforms features from the range of `0 to +1`

$$x_{transformed} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```
% Create a M-file named "normalize.m"

% Function for Normalized Feature Scaling
function x_transformed = normalize(feature)
    % mean of the features
    min_feature = min(feature);
    % standard deviation of features
    max_feature = max(feature);
    % standardize
    x_transformed = (feature - min_feature)/(max_feature - min_feature);
end
```

```
% standardize feature scaling for age
standardized_age = standardize(data4.Age);
data4.Age = standardized_age;

% standardized feature scaling for annual salary
standardized_salary = standardize(data4.("Annual Salary"));
data4.("Annual Salary") = standardized_salary;

% view the data
data4
```

data4 = 10×4 table

	Location	Age	Annual Salary	Opinion
1	'US'	0.31675	0.71102	'not liked'
2	'Asia'	-1.5106	-1.3644	'not liked'
3	'Africa'	-0.90152	-0.84552	'not liked'
4	'Asia'	-0.29238	-0.2402	'not liked'
5	'Africa'	0.80405	-6.0532e-05	'Liked'
6	'US'	-0.53604	-0.49962	'Liked'
7	'Asia'	-0.048731	-1.0185	'not liked'
8	'US'	0.31675	1.3163	'Liked'
9	'Africa'	2.1441	1.6622	'not liked'
10	'US'	-0.29238	0.27865	'Liked'

```
% Normalize features scaling for age
normalized_age = normalize(data4.Age);
data_norm =data4;
data_norm.Age = normalized_age;

% Normalize features scaling for annual salary
normalized_salary = normalize(data4.("Annual Salary"));
```

```
data_norm.("Annual Salary") = normalized_salary
```

```
data_norm = 10×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	0.5	0.68571	'not liked'
2	'Asia'	0	0	'not liked'
3	'Africa'	0.16667	0.17143	'not liked'
4	'Asia'	0.33333	0.37143	'not liked'
5	'Africa'	0.63333	0.45077	'Liked'
6	'US'	0.26667	0.28571	'Liked'
7	'Asia'	0.4	0.11429	'not liked'
8	'US'	0.5	0.88571	'Liked'
9	'Africa'	1	1	'not liked'
10	'US'	0.33333	0.54286	'Liked'

Handling Outlier

An outlier is a value that has huge/abnormal deviation from other values. It is important to get rid of the outliers in data before passing it to the Machine Learning algorithm.

- Use `isoutlier()` to check if there is an outlier.
- **How to:** Delete Outlier row, Fill outlier.

Techniques for handling outliers

- Calculate **Z-Score**

$$Z = \frac{x-\mu}{\sigma}$$

μ represents mean, σ represents standard deviation. **N.B:** If the data is deviated more/less than 3-standard deviation, it is possibly an outlier.

- Median Absolute Deviation

$$\text{MAD} = \text{median}(|x - \text{median}(x)|)$$

By default, an outlier is a value that is more than three scaled median absolute deviation (MAD) away from the median.

```
data5 = readtable("..\data\data_5.csv","VariableNamingRule","preserve");
data5(1:3,:)
```

```
ans = 3×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'

	Location	Age	Annual Salary	Opinion
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'

```
% check for outliers
% it is applied to numerical data only
sum(isoutlier(data5.Age))
```

```
ans =
     0
```

```
% check the outlier
data5.Age(isoutlier(data5.Age)>0)
```

```
ans =

   0×1 empty double column vector
```

How MATLAB detect an Outlier?

By default, MATLAB uses the information of the `median of a data` to determine its outlier. It will discard values **3times** greater or lesser from the median.

```
% Outliers by default calculated using mean
isoutlier()

% Outliers using arguments/method
isoutlier(numerical_data,"method","mean")
```

The methods are:

- mean
- median
- movmean

Deleting Outliers

```
% delete outlier using logical indexing
outliers = isoutlier(data5.Age);

data5 = data5(~outliers,:)
```

```
data5 = 10×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	63777	'Liked'
6	'US'	33	58000	'Liked'

	Location	Age	Annual Salary	Opinion
7	'Asia'	37	52000	'not liked'
8	'US'	40	79000	'Liked'
9	'Africa'	36	83000	'not liked'
10	'US'	35	67000	'Liked'

Filling Outliers (Recommended)

The fill outlier methods are:

- central
- linear
- previous
- next and so on.

```
data_fill = readtable("..\data\data_5.csv","VariableNamingRule","preserve");

% fill with center -> median
Age = filloutliers(data_fill.Age,"center");

% the default is median, you can specify using the 3rd argument.
data_fill.Age = Age
```

```
data_fill = 10×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	63777	'Liked'
6	'US'	33	58000	'Liked'
7	'Asia'	37	52000	'not liked'
8	'US'	40	79000	'Liked'
9	'Africa'	36	83000	'not liked'
10	'US'	35	67000	'Liked'

Encoding Categorical data

Since the machine learning models are based on mathematical formula, the categorical data are conveted into numerical data.

```
data_fill
```

```
data_fill = 10×4 table
```

	Location	Age	Annual Salary	Opinion
1	'US'	40	72000	'not liked'
2	'Asia'	25	48000	'not liked'
3	'Africa'	30	54000	'not liked'
4	'Asia'	35	61000	'not liked'
5	'Africa'	44	63777	'Liked'
6	'US'	33	58000	'Liked'
7	'Asia'	37	52000	'not liked'
8	'US'	40	79000	'Liked'
9	'Africa'	36	83000	'not liked'
10	'US'	35	67000	'Liked'

You cannot assign any number value to represent a `nominal` categorical data but you can for an `ordinal` categorical data.

For instance, if Us = 0, Asia = 1, and Africa = 2. The machine learning model would assume that **`Africa > Asia > Us`** but for ordinal categorical data; poor = 1, good = 2, excellent = 3. That is, **`excellent > good > poor`**.

Another example of the ordinal data is grades: **A, B, C, D, E, F**. A = 5, B = 4, C = 3, D = 2, E = 1, F = 0.

```
% manual encoding with inputting values on excel
encoding = readtable("..\data\encode.csv","VariableNamingRule","preserve")
```

```
encoding = 10×6 table
```

	Age	Annual Salary	Opinion	Africa	Asia	Us
1	40	72000	'not liked'	0	0	1
2	25	48000	'not liked'	0	1	0
3	30	54000	'not liked'	1	0	0
4	35	61000	'not liked'	0	1	0
5	44	63777	'Liked'	1	0	0
6	33	58000	'Liked'	0	0	1
7	37	52000	'not liked'	0	1	0
8	40	79000	'Liked'	0	0	1
9	36	83000	'not liked'	1	0	0
10	35	67000	'Liked'	0	0	1

Implementing Encoding of Nominal data type with Matlab Algorithm

- MATLAB provides a function `ismember()`

```
% To check if a datatype is a member
mem = ismember(data5.Location,'Asia');

% convert the logical value to double
mem_arr = double(mem);

% convert the array to table
% N.B: The variable name must be converted to cell
mem_table = array2table(mem_arr,'VariableNames',{'France'})
encoded_data = categorical_encoding(data5,data5.Location)

% delete the 'Location' feature since it has been encoded
encoded_data.Location = []
```

N.B: Having encoded the Location column, the next categorical data is the **Opinion.** Make sure the variables fulfil the convention of variable naming.

- No space between compound words; use snake or camel case instead.
- Special charater or number shouldn't start the naming

```
% save the encoded_data to modify the opinion to fulfil conventional variable naming
```

```
writetable(encoded_data,"encoded_data.csv")
```

```
% notice the hyphen in between 'not_liked'
data_en = readtable('encoded_data.csv','VariableNamingRule','preserve')
```

Implementing Encoding of Ordinal data type with Matlab Algorithm

```
% read the data
data6 = readtable("../data/data_6.csv",'VariableNamingRule','preserve')
```

data6 = 10×4 table

	Refund	Marital Status	Yearly Income	Cheat
1	'Yes'	'Single'	'High'	'No'
2	'No'	'Married'	'Very High'	'No'
3	'No'	'Single'	'Average'	'No'
4	'Yes'	'Married'	'High'	'No'
5	'No'	'Divorced'	'Low'	'Yes'
6	'No'	'Married'	'Low'	'No'
7	'Yes'	'Divorced'	'High'	'No'
8	'No'	'Single'	'Average'	'Yes'
9	'No'	'Married'	'Average'	'No'
10	'No'	'Single'	'Average'	'Yes'

```
% encode the yearly income
```

```
yearlyIncome_encode = categorical_ordinal_encode(data6.("Yearly Income"),{'Low','Average','High
yearlyIncome_encode', data6.("Yearly Income")
```

```
ans = 1×10
     3     5     2     3     1     1     3     2     2     2
ans = 10×1 cell
'High'
'Very High'
'Average'
'High'
'Low'
'Low'
'High'
'Average'
'Average'
'Average'
```

```
data6_copy = data6;
data6_copy.("Yearly Income") = yearlyIncome_encode
```

```
data6_copy = 10×4 table
```

	Refund	Marital Status	Yearly Income	Cheat
1	'Yes'	'Single'	3	'No'
2	'No'	'Married'	5	'No'
3	'No'	'Single'	2	'No'
4	'Yes'	'Married'	3	'No'
5	'No'	'Divorced'	1	'Yes'
6	'No'	'Married'	1	'No'
7	'Yes'	'Divorced'	3	'No'
8	'No'	'Single'	2	'Yes'
9	'No'	'Married'	2	'No'
10	'No'	'Single'	2	'Yes'

Function for encoding Nominal categorical data.

```
% Categorical data to dummy variables
function data = categorical_encoding(data,variable)
% To encode categorical data
    unique_values = unique(variable);

    for i=1:length(unique_values)
        encode_variable(:,i) = double(ismember(variable,unique_values{i}));
    end

    T = table;
    [rows, col] = size(encode_variable);

    for i=1:col
        T1 = table(encode_variable(:,i));
        T1.Properties.VariableNames = unique_values(i);
```

```
            T = [T T1];
        end

        data = [T data];
    end
```

Function for encoding ordinal categorical data.

```
    function new_var = categorical_ordinal_encode(variable,values_set,numbers)

        [rows,col] = size(variable);


        new_var = zeros(rows,1);

        for i=1:length(values_set)
            indices = ismember(variable,values_set{i});
            new_var(indices) = numbers(i);
        end

    end
```

Count Fequency Encoding

It can also be used to handle nominal categorical data using the frequency of occurence.

```
data6
```

data6 = 10×4 table

	Refund	Marital Status	Yearly Income	Cheat
1	'Yes'	'Single'	'High'	'No'
2	'No'	'Married'	'Very High'	'No'
3	'No'	'Single'	'Average'	'No'
4	'Yes'	'Married'	'High'	'No'
5	'No'	'Divorced'	'Low'	'Yes'
6	'No'	'Married'	'Low'	'No'
7	'Yes'	'Divorced'	'High'	'No'
8	'No'	'Single'	'Average'	'Yes'
9	'No'	'Married'	'Average'	'No'
10	'No'	'Single'	'Average'	'Yes'

```
tabulate(data6.Refund);
```

```
  Value    Count   Percent
    Yes        3     30.00%
     No        7     70.00%
```

```
[row, col] = size(tabulate(data6.Refund));
```

```
size(data6.Refund,1);

% algorithm

freq_encode = frequency_count_encoder(data6,data6.Refund,"RefundEncode")
```

```
freq_encode = 10×5 table
```

	Refund	Marital Status	Yearly Income	Cheat	RefundEncode
1	'Yes'	'Single'	'High'	'No'	3
2	'No'	'Married'	'Very High'	'No'	7
3	'No'	'Single'	'Average'	'No'	7
4	'Yes'	'Married'	'High'	'No'	3
5	'No'	'Divorced'	'Low'	'Yes'	7
6	'No'	'Married'	'Low'	'No'	7
7	'Yes'	'Divorced'	'High'	'No'	3
8	'No'	'Single'	'Average'	'Yes'	7
9	'No'	'Married'	'Average'	'No'	7
10	'No'	'Single'	'Average'	'Yes'	7

Further Learning:

- Matlab Machine Learning Onramp. link
- Udemy: Go from Beginner to Expert in MATLAB
- Youtube: Data preprocessing in MATLAB