A DSL to Automate Initial and Exploratory Analysis for Regression-Based Data Analytics Projects

Sayed Ahmed and Sepehr Bayat *

Contents

1 Abstract

Model Driven Development (MDD) can facilitate faster software development while providing consistency and protecting code integrity. Domain Specific Languages utilizing MDD can simplify the development of a specific set of repetitive tasks for a particular aspect and domain. In machine

*(alphabetical order)

learning and data analytics projects, several tasks such as data cleaning, data adjustment, Exploratory Data Analysis (EDA), and Initial Data Analysis (IDA) are common for most projects. Although the extent and need can vary, the code for several aspects can be made the same to avoid repetitive code and tasks. Hence, in this project, we have developed a Domain Specific Language utilizing MDD. We have developed a Domain Specification Language (DSL) to automate Initial and Exploratory Analysis for Regression-Based Data Analytics Projects. Our project automates and writes code for Data Cleaning, Initial Analysis, and Exploratory Data Analysis. Anyone wanting to write a Data Analytics Project can use this DSL. The user can use a GUI or Text Format to define data sources and relationships among data sources. Relationships can be selected from predefined selections. The developer can select the types of analysis he or she wants such as clean/replace null values, unitary analysis, unitary plots, bi-variate regression, bi-variate plots, and r-square calculation. The DSL, by default, can/will generate code for a pre-configured (default configuration) set of tasks where the developer does not want to provide a custom list of tasks. Mandatory requirements are that our DSL will write code to clean/replace null/missing values, align data type to the majority type, r-square/adjusted r-square calculation, data normalization, unitary plot, bi-variate plot, bi-variate regression plot with the target.

2 Introduction

Tools to Use

- Emfatic/Ecore
- Eugenia
  - Xtext in Eugenia
- Papyrus (UML tool)
- Probably: Model querying (EOL)
- Probably: Model validation (EVL)

3 Related Work

4 Proposed Solution

4.1 Methodology Overview

4.2 Overall Steps

To develop the DSL, the steps that we followed are:

1. Develop a Metamodel for the DSL to automate Data Cleaning/EDA/IDA
   - Define the abstract - syntax i.e. define the terminology: Abstract Syntax
   - Create/define Concrete Syntax: Relationship/Symbols utilizing the Abstract Syntax
   - Define the semantic syntax
2. Develop Model: Models are instances of Meta Models

3. Develop UI/Editor for the developer (use GMF, Eugenia): GMF/Graphical UI to write program.

4. Model to code

While creating the Meta Models we will define some rules so that it can verify the validity of the metamodel. These rules will verify the flow and steps that developers use for a project. We may provide developers the facility to create a state diagram to define the steps of the code flow. These rules can verify the validity of such state flow diagrams.

We will use Eclipse Ecore, and EMFatic for creating Meta Models. We also have come across tools from Microsoft as part of Visual Studio that is also an option to utilize.

4.3 Artefacts of our DSL

Our DSL can result in the following outputs:

1. Null Value Replaced Data
2. Missing Value Replaced Data
3. Rescaled, harmonized, and Normalized data
4. R-Square Analysis with P-values and F-Significance
5. Visualizations for Unitary, Bi-Variate, and multivariate analysis
6. Correlation plots (Pearson, Spearman, Kendall Tau Coeff)

Overall, the output will have data, code, and visualizations. As an example, our DSL can provide some features like excel plugins named XL-STAT, and Data Analysis Toolpak. However, we will generate code as well that developers will be able to modify.

Output/Artefacts in Chronological Order:

1. Null Value Replaced Data
2. Missing Value Replaced Data
3. Rescaled, harmonized, and Normalized data
4. Visualizations for Unitary, Bi-Variate, and multivariate analysis
5. R-Square Analysis with P-values and F-Significance
6. P-value, F-significance, and some comments: important/not-important factors/Variables.
7. Correlation plots (Pearson, Spearman, Kendall Tau Coeff)

4.4 Models and State Diagrams

From the meta-model, models can be created. Developers will have the facility to create models based on the meta-model. Developers will create the models using abstract, and concrete syntax. We have provided primarily text-based Model creation. However, graphics based model creation is an option as well.

Potential models that developers might be able to create are as follows:

- NullMissingModel
- ScaleModel
- NormalizeModel
- StandardizeModel
- VisualPlotModel with Unary/Binary/Multi Sub Models
- RSquareModel
- PValueFSignificanceModel

Developers will have the option to create a state flow diagram that will utilize the model to define the work/code-generation flow they want. State diagrams will contain conditional/decisional flows.

4.5 Meta Model Details

A diagram of our meta model is provided in the figure 1. Meta Model Details are provided below:

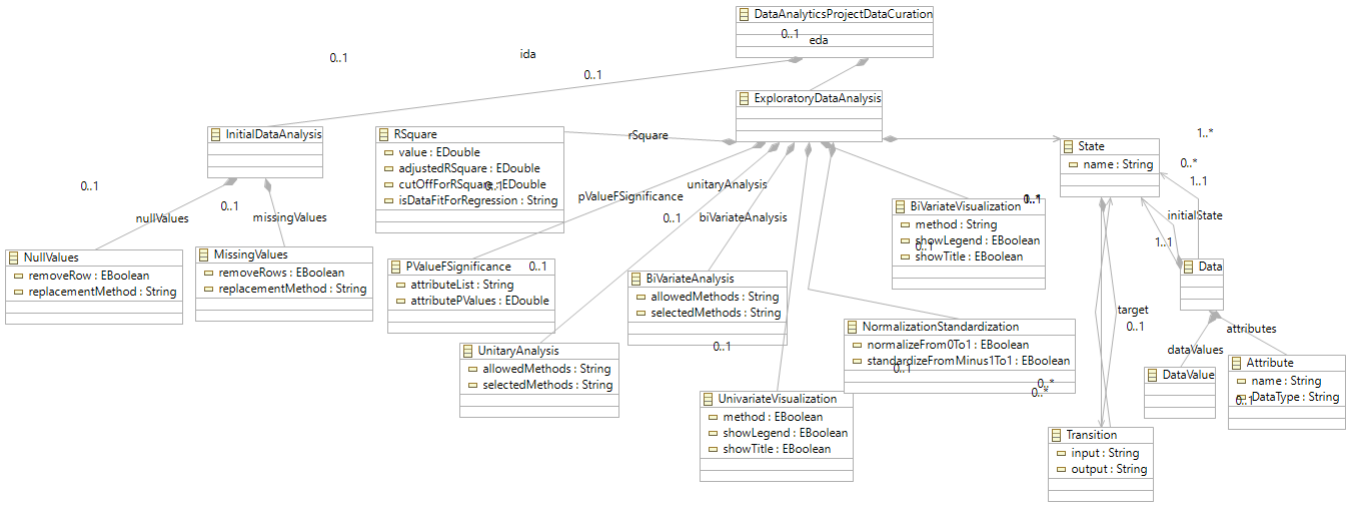


Figure 1: Data Curation Meta Model

Abstract Syntax

- Null
- Missing

- RSquare
- PF
- UVisualize
- BiVisualize

Concrete Syntax Elements

- Box: Model
- Circle: Process
- Rhombus: Start and End.
- Directed arrow: flow

Symantic Syntax Elements

- to be filled

Meta Model Rules:

- No visualizations before Normalization/Standardization unless excepted
- All independent features will need to be numeric, else drop/convert
- Target feature will need to be numeric

Meta Model Need and Approaches:

Do we need metamodels? or we can just directly start with Models? Because of the context, applications will have varying needs for Data Cleaning, EDA, and IDA hence, Metal Model will be advantageous to support these varieties. Meta models will provide the flexibility to extend this project to other areas such as Natural Language Processing (NLP), Classification, and Clustering projects. Meta-models can help in one of two approaches such as one as we mentioned that models can be created for Regression, Classification, Clustering, and NLP projects; another way metal models help is: among all the functionalities/codes/features that meta-models will provide, a project can choose to use a sub-set of them; hence, creating a variation of the models. Hence, meta-models will be important. Selection can be automated or developer selected.

Regression, Classifications, clustering, and NLP all need EDA/IDA; hence, metal-models can bring the common EDA/IDA features/code into the meta models.

Similarity:

Musicians: Characteristics and variations

EDA/IDA: Models with varying characteristics. We can create variations.

4.6 EMFatic Meta Model: Abstract Syntax

Class EDA {

- attr name: string
- attr style: style-category
- Null
- Missing
- Normal
- Scale
- standard
- unary
- binary

}

Class NULL {

- attr name: string

}

Class Normal {

- attr name: string

}

enum style-category {

- attr name: string

}

4.7 Models from EMFatic Meta Model

Class EDA-REG {

- attr name: string

}

Class IDA-REG {

- attr name: string

}

Data
Null Replace
Scale

Data
Null Replace
Scale
Unary Visual
R-Square

4.8 Graphical Concrete Syntax Example

4.9 Screenshots of Metamodels from ongoing implementation

Provided in figure 2

4.10 Flow IDA/EDA in a Diagram

4.11 UI Editor

Use GMF/Eugenia to create the editor (make use of .gmfgen and .diagram files)

From class notes: "• Once the .gmfgraph, .gmftool, and .gmfmap models are in place, GMF provides a model-to- model transformation that creates a generator model from them (.gmfgen)"
"

5 Experiments Changed

6 Results and Discussion

7 Conclusion

7.1 Future Work

References

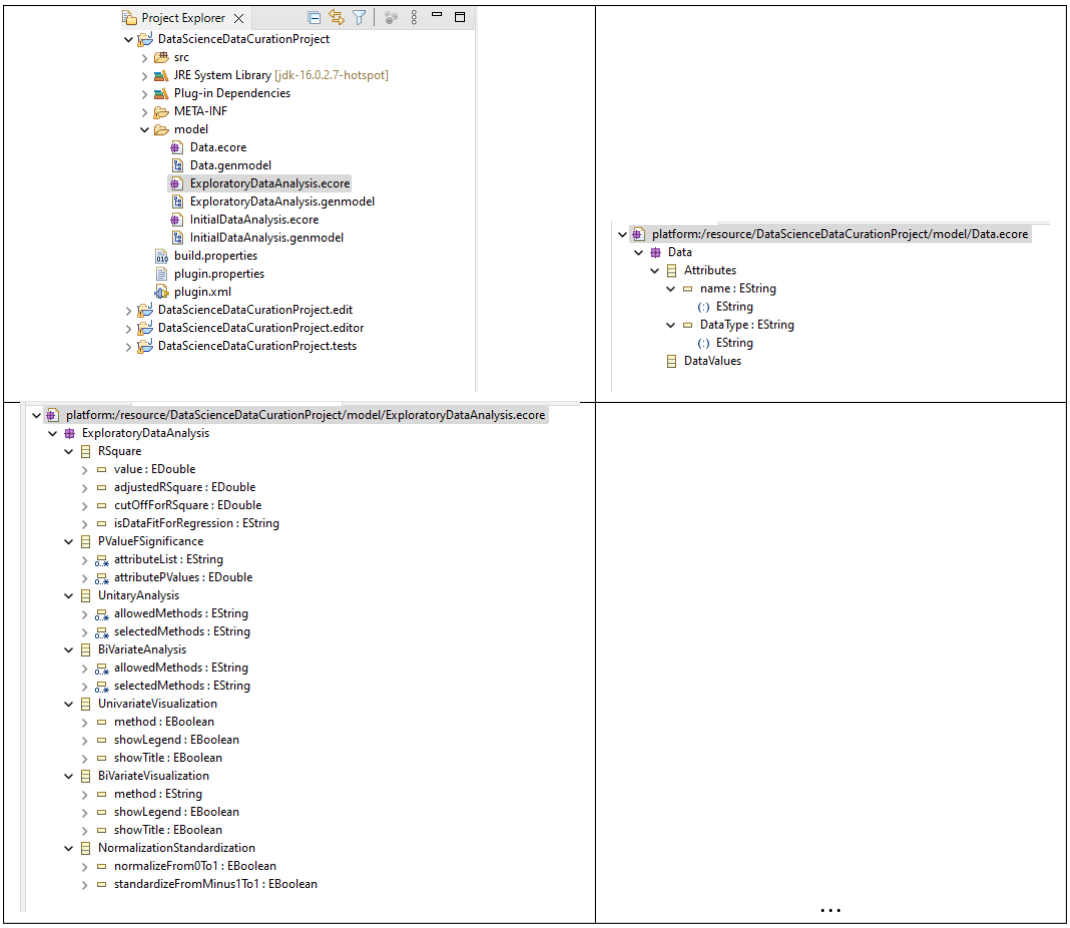
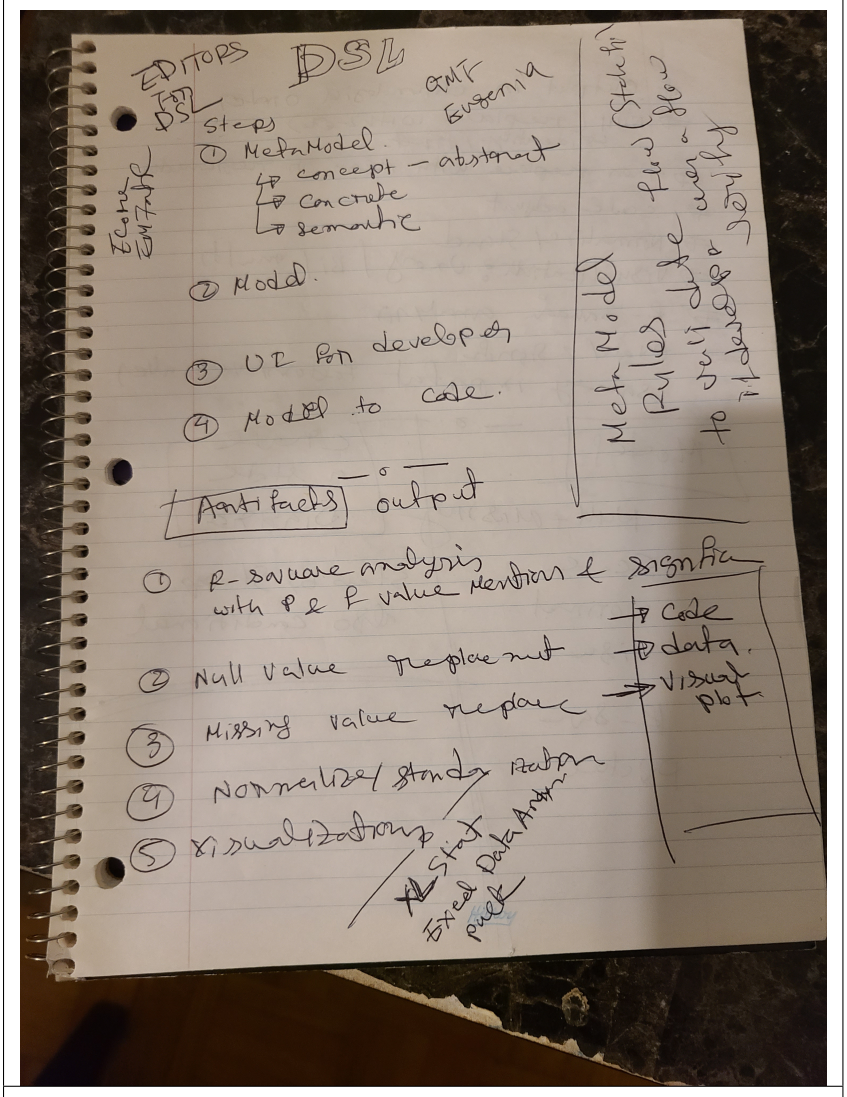
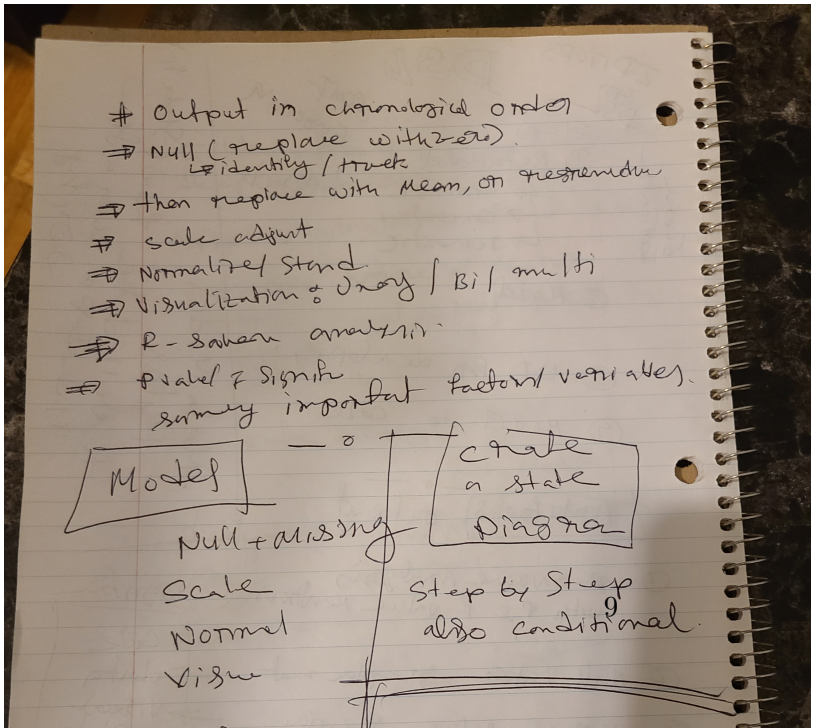


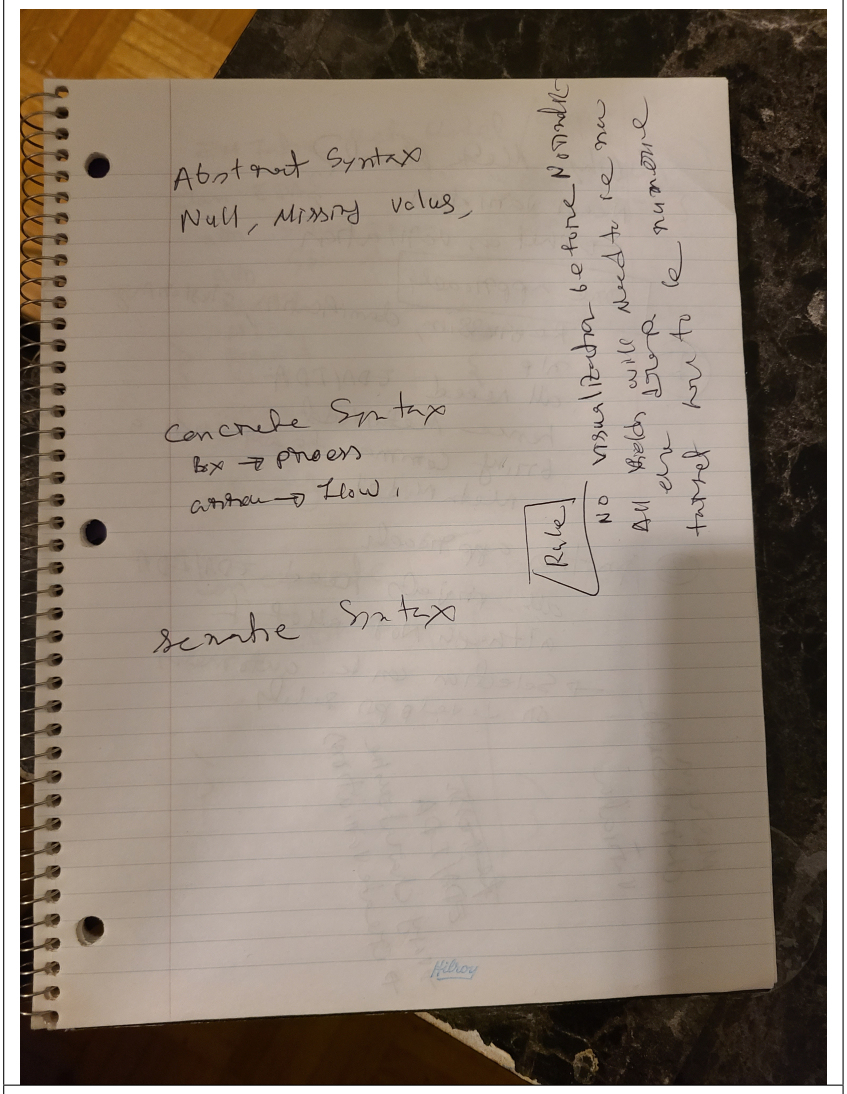
Figure 2: Screenshots of Metamodels from ongoing implementation

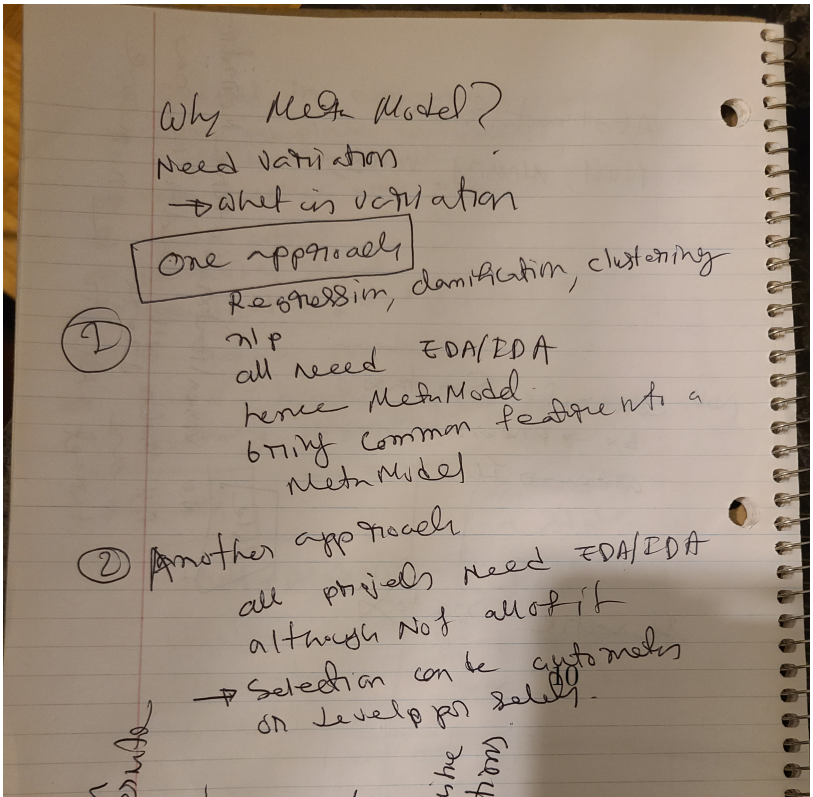
Figure 3: Flow EDA/IDA

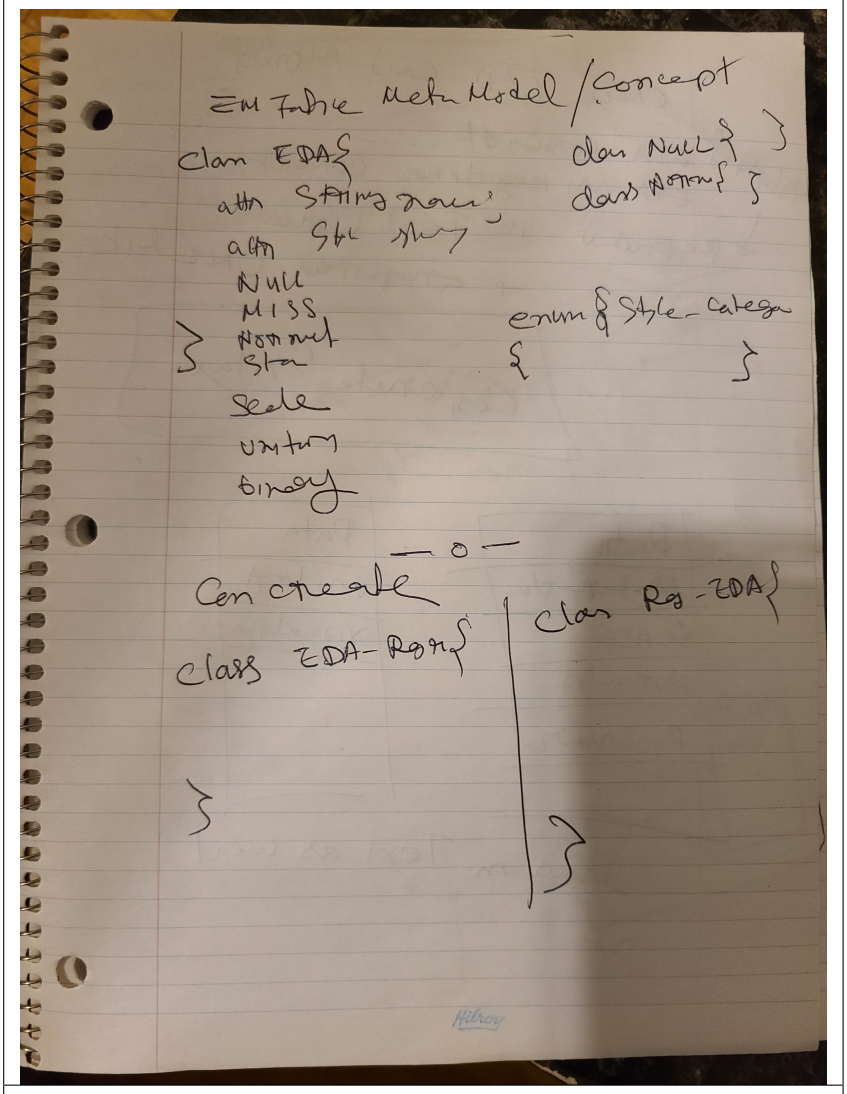
8 Appendix

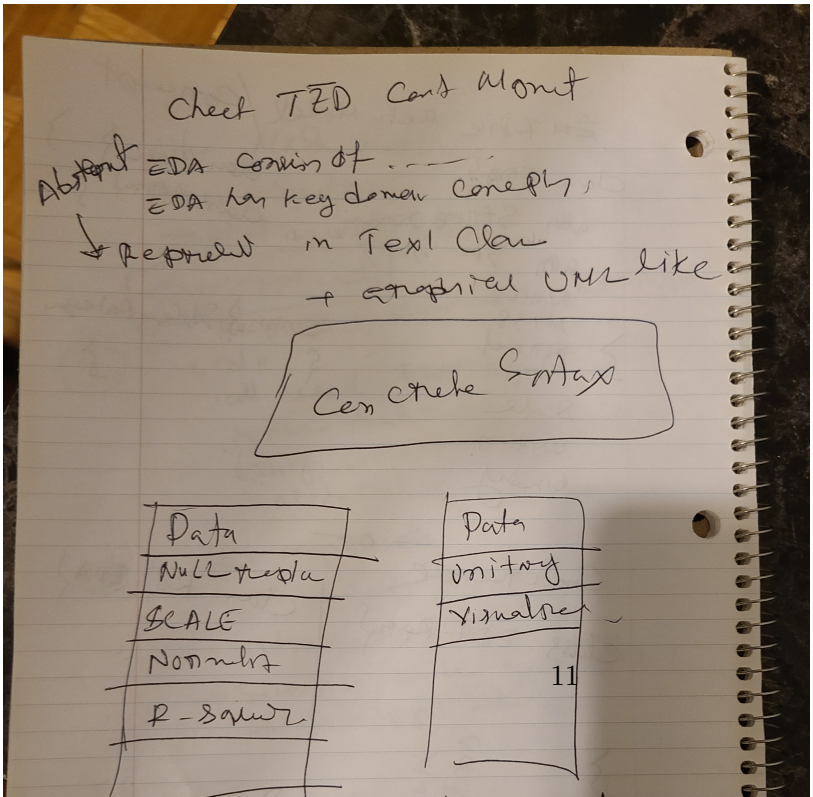

EDITORS
DSL
Steps
① MetaModel.
concept – abstract
Concrete
Semantic
② Model.
③ UI for developer
④ Model to code.
Artifacts
output
① R-square analysis with P & F value mention & signifi
② Null value replacement
③ Missing value replace
④ Normalize/ standardization
⑤ Visualization
Code
data.
Visual plot
Fixed Data Analysis pack

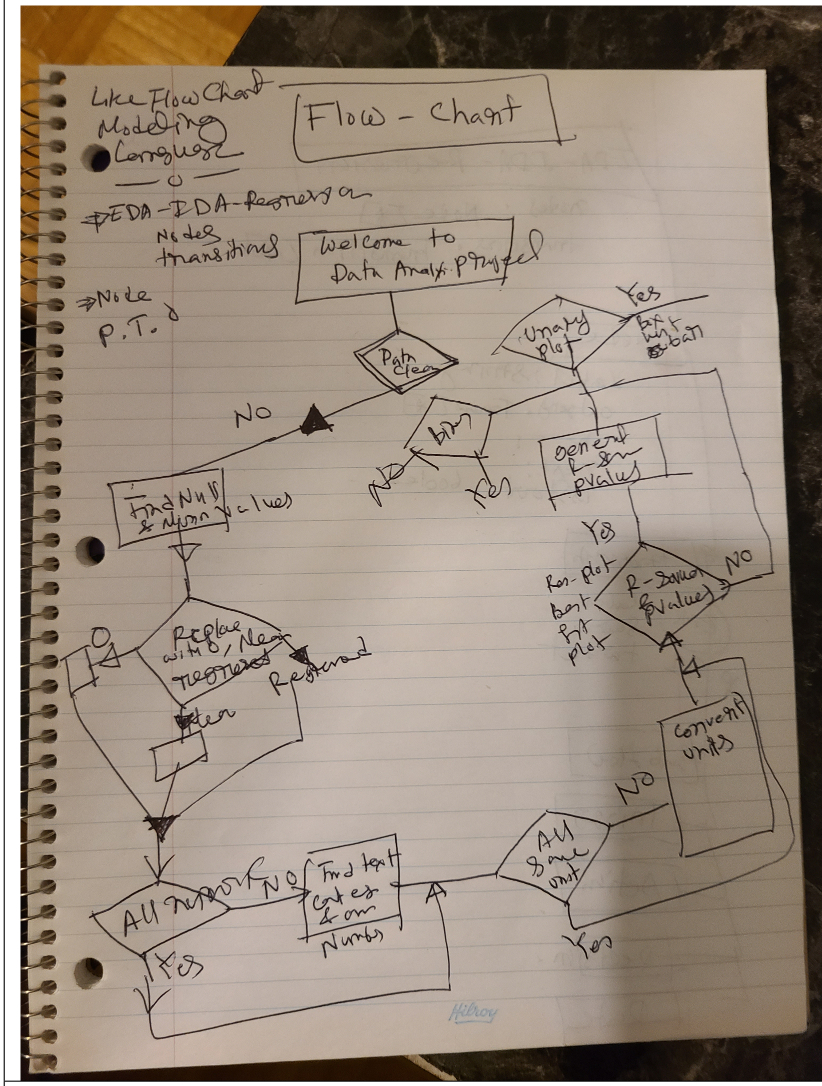

Output in chronological order
Null (replace with zero)
identify / track
then replace with mean, or regression
scale adjust
Normalize/ Stand
Visualization : Univ / Bi / multi
R-Square analysis
P value & Signifi
survey important factors/ variables
Model
Null + Missing
Scale
Normal
Visu
create a state Diagram
Step by Step
also conditional

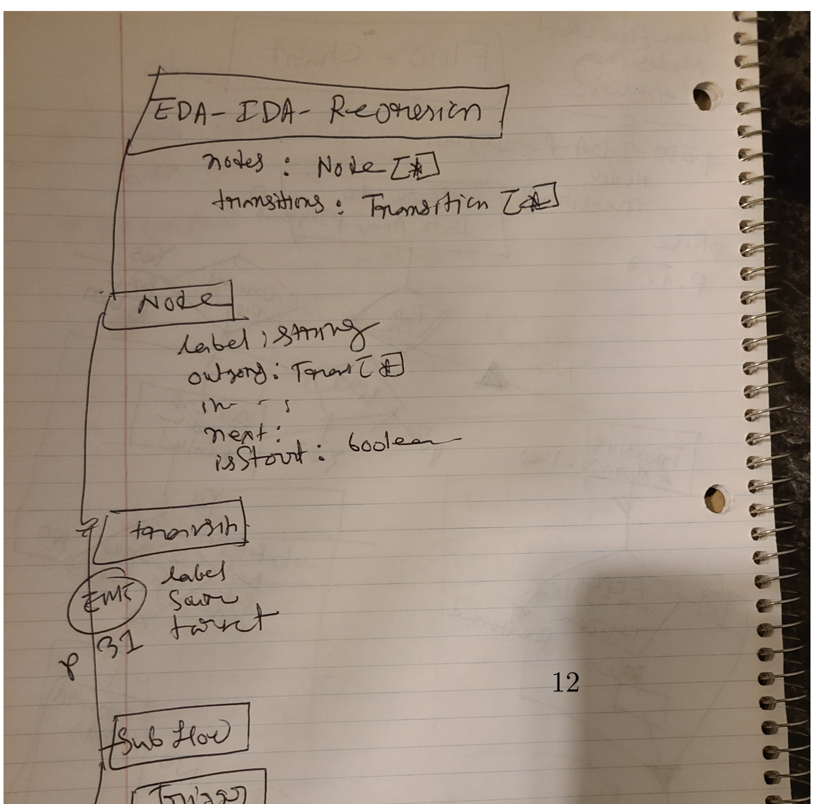

Abstract Syntax
Null, Missing value,
Concrete Syntax
Rule
NO
Scrabe Syntax


Why Meta Model?
Need variation
→ what is variation
One approach
Regression, classification, clustering
(1)
(2)


EM Table Meta Model / Concept
class EDA{
attr String name;
attr Str
NULL
MISS
Nominal
Stra
Scale
Unitary
binary
class Null{ }
class Nominal{ }
enum{ Style_Catego
{ }
Concrete
class EDA-Rang{
}
class Ra-EDA{
}


Check TED Card Mount
Abstract EDA consist of....
EDA has key domain concepts,
↳ represent in Text Class
+ graphical UML like
Concrete Syntax
Data
NULL-redu
SCALE
Nominal
R-Solver
Data
Unitary
Visualiser
11


Flow - Chart
Yes
No
Find text
Convert units
All Same unit
Yes


EDA-IDA-Regression
nodes : Node [*]
transitions : Transition [*]
Node
label : String
isStart : boolean
Sub Flow