HORIZON 2020

SmartCLIDE

Deep Learning Engine(DLE)

AIR INSTITUTE

SmartCLIDE Main Concepts

❖ Service-Oriented Systems

Modern applications compose available web services to build new software products. In this context, service-oriented systems' main tasks can fall into four steps:

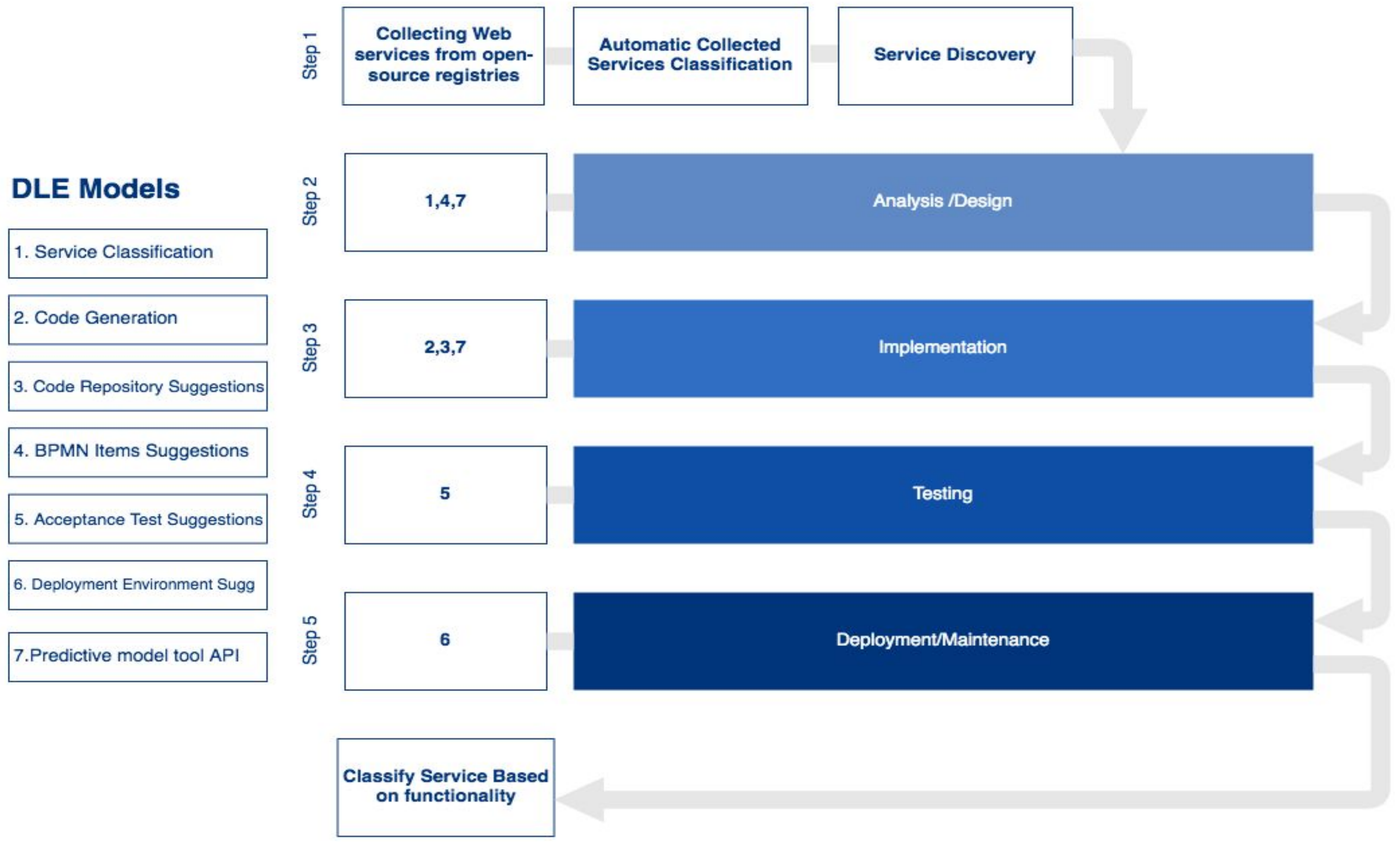
1. **Identifying system requirements**
2. **Finding and discovering service registries and providing a pool of services**
3. **Classifying the discovered services to identify a list of candidate services with the same functionality for particular tasks**
4. **Ranking selected services with the same functionality**

❖ Integrated Development Environment(IDE)

IDEs could have more automation and intelligence to help developers. These features can be obtained by using Artificial Intelligence (AI) and Machine Learning techniques. Most IDEs include several tools to cover most aspects of software development like analyzing, designing, implementing, testing, documenting, and maintaining .
To increase intelligence, these IDEs have embedded training models into modern versions.
This task can fall into two methods:

1. **Improving current functionalities(e.g,Code/Item suggestion)**
2. **Adding new functionalities(AI-Based code generation)**

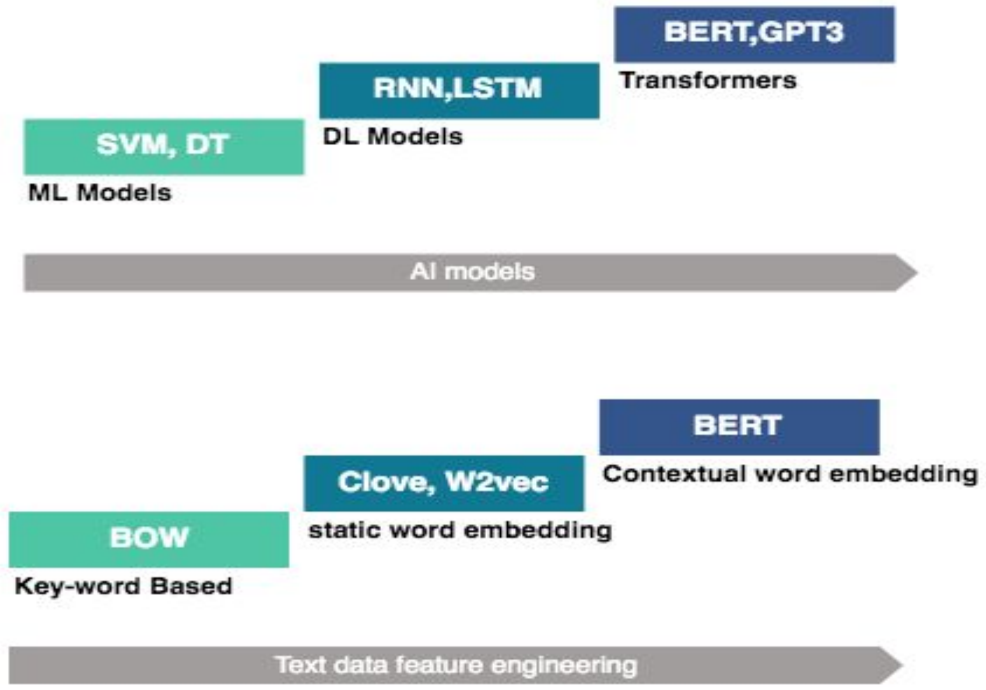
SmartCLIDE DLE Models



Step 1
Collecting Web services from open-source registries
Automatic Collected Services Classification
Service Discovery
DLE Models
1. Service Classification
2. Code Generation
3. Code Repository Suggestions
4. BPMN Items Suggestions
5. Acceptance Test Suggestions
6. Deployment Environment Sugg
7.Predictive model tool API
Step 2
1,4,7
Analysis /Design
Step 3
2,3,7
Implementation
Step 4
5
Testing
Step 5
6
Deployment/Maintenance
Classify Service Based on functionality

DLE Algorithms/Datasets

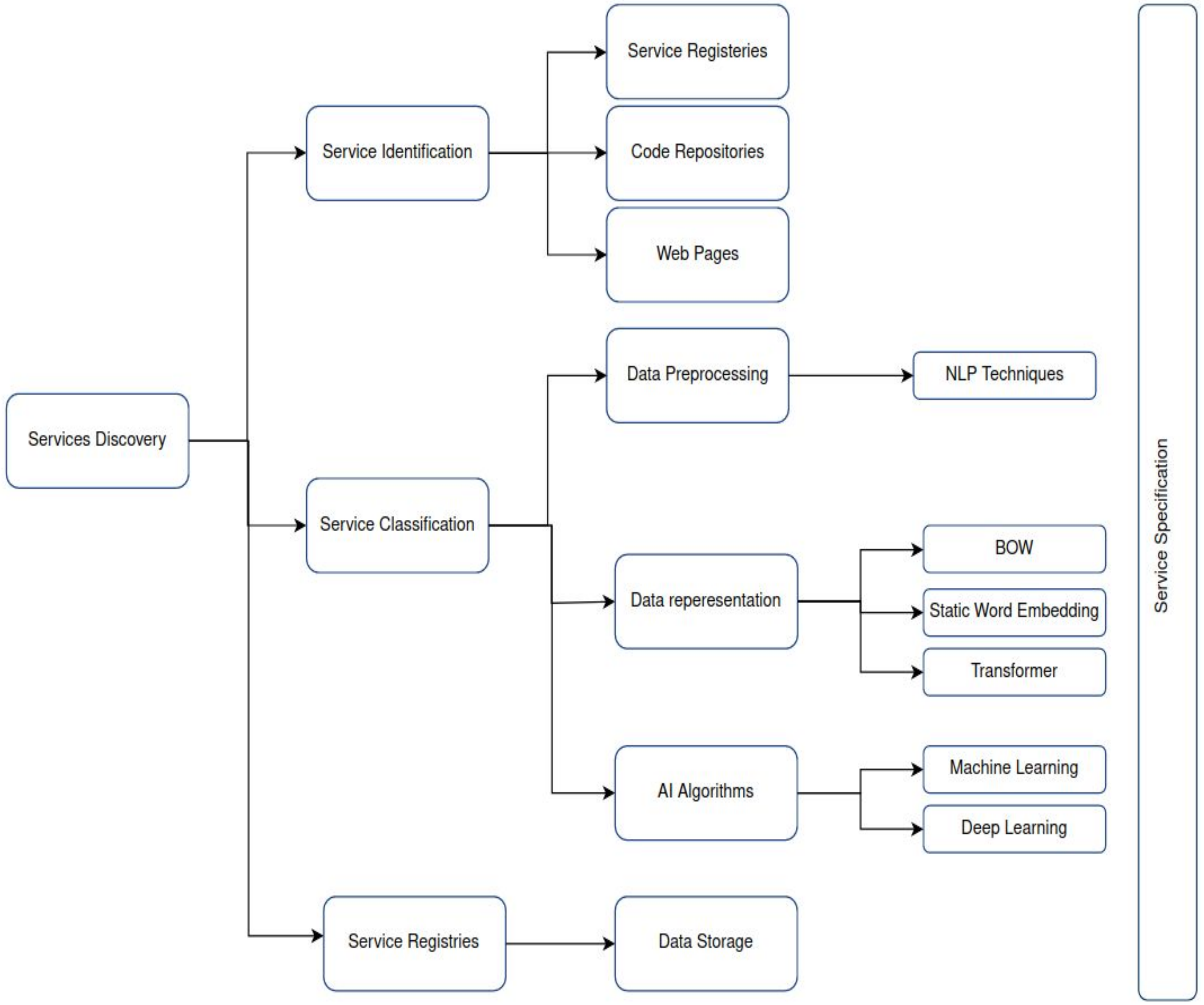
AI-based Text Processing

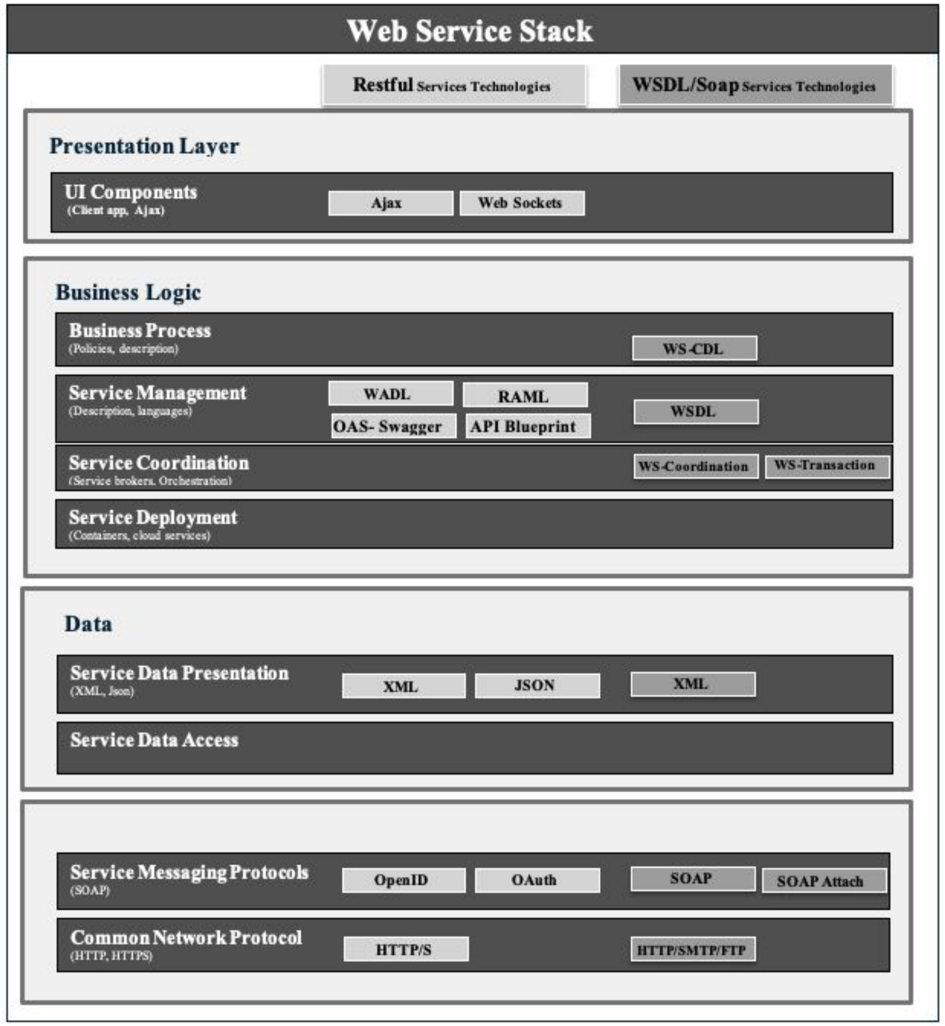


DLE Top Datasets

Dataset	Rows	Sources
Programableweb	24456	[1]
BPMN Dataset	300+	-
GitHub Java Corpus	14875	[2]
CodeSearchNet	496688	[3]

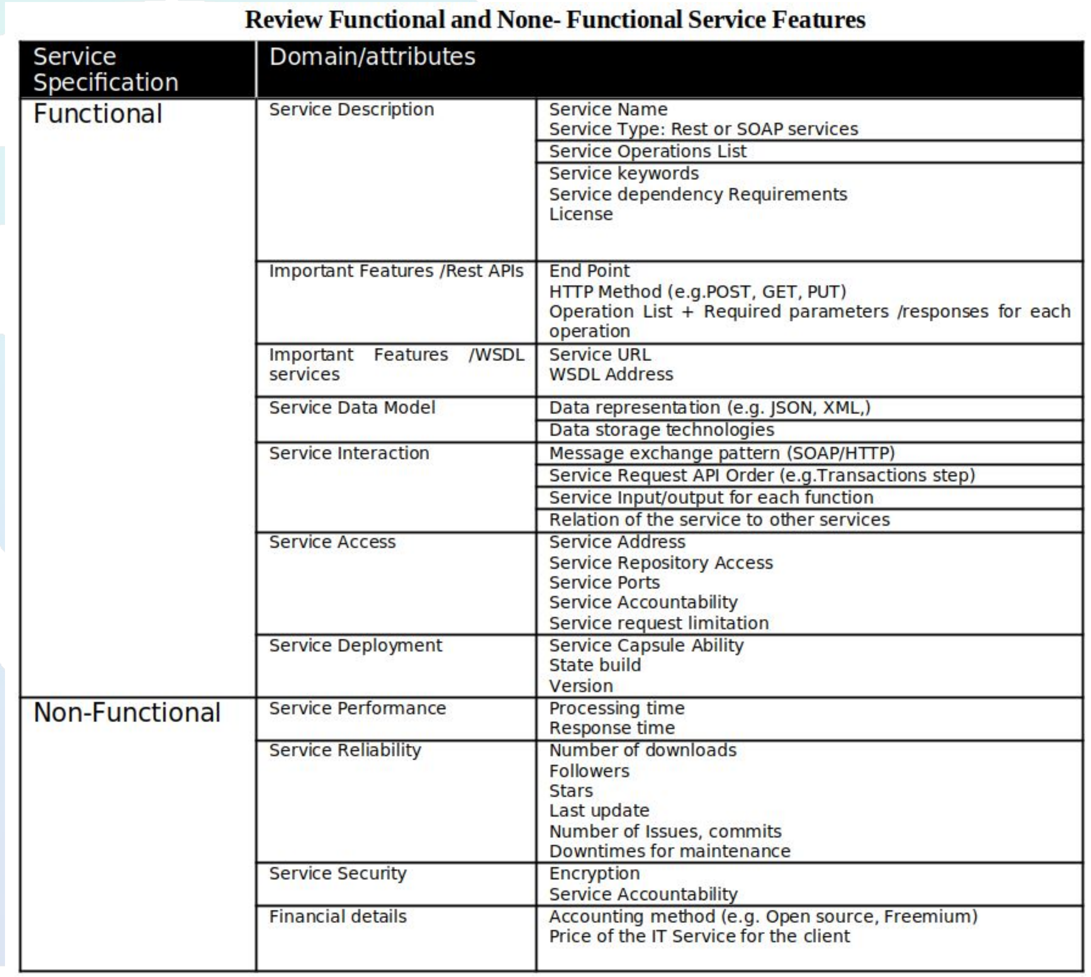
Service Specification





Service Specification

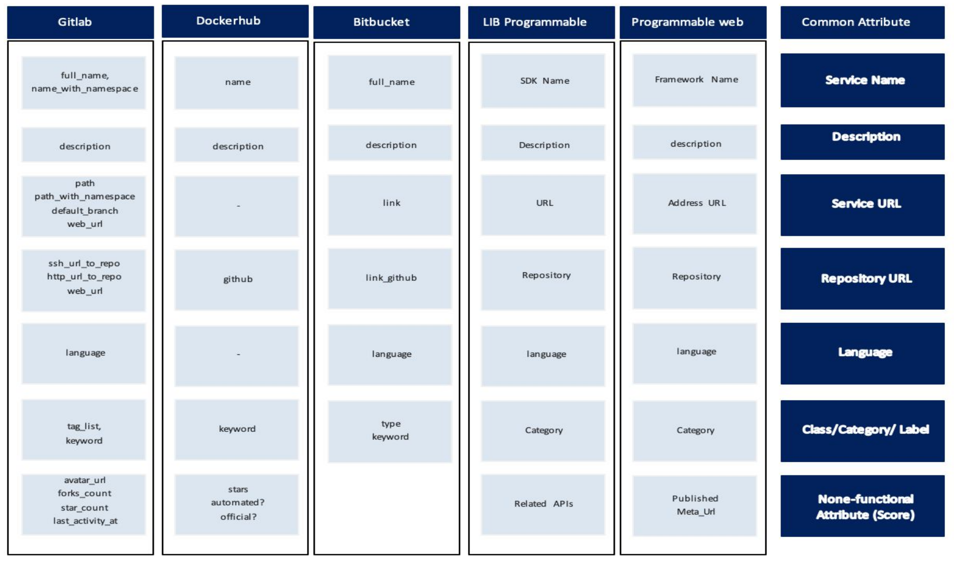
Extractable Features From the State Of the Art



Review Functional and None- Functional Service Features

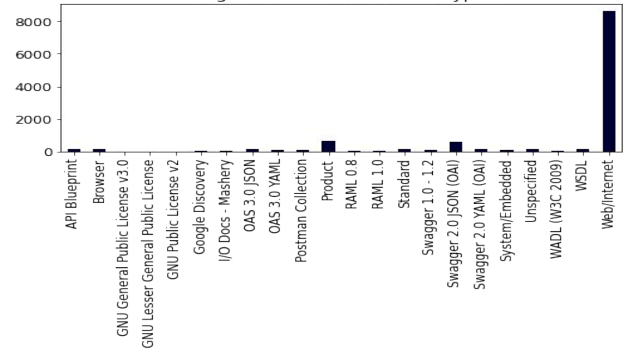
Service Specification	Domain/attributes	
Functional	Service Description	Service Name Service Type: Rest or SOAP services
		Service Operations List
		Service keywords Service dependency Requirements License
	Important Features /Rest APIs	End Point HTTP Method (e.g.POST, GET, PUT) Operation List + Required parameters /responses for each operation
	Important Features /WSDL services	Service URL WSDL Address
	Service Data Model	Data representation (e.g. JSON, XML,)
		Data storage technologies
	Service Interaction	Message exchange pattern (SOAP/HTTP)
		Service Request API Order (e.g.Transactions step)
		Service Input/output for each function
		Relation of the service to other services
	Service Access	Service Address Service Repository Access Service Ports Service Accountability Service request limitation
	Service Deployment	Service Capsule Ability State build Version
Non-Functional	Service Performance	Processing time Response time
	Service Reliability	Number of downloads Followers Stars Last update Number of Issues, commits Downtimes for maintenance
	Service Security	Encryption Service Accountability
	Financial details	Accounting method (e.g. Open source, Freemium) Price of the IT Service for the client

Extractable Features From Online Sources

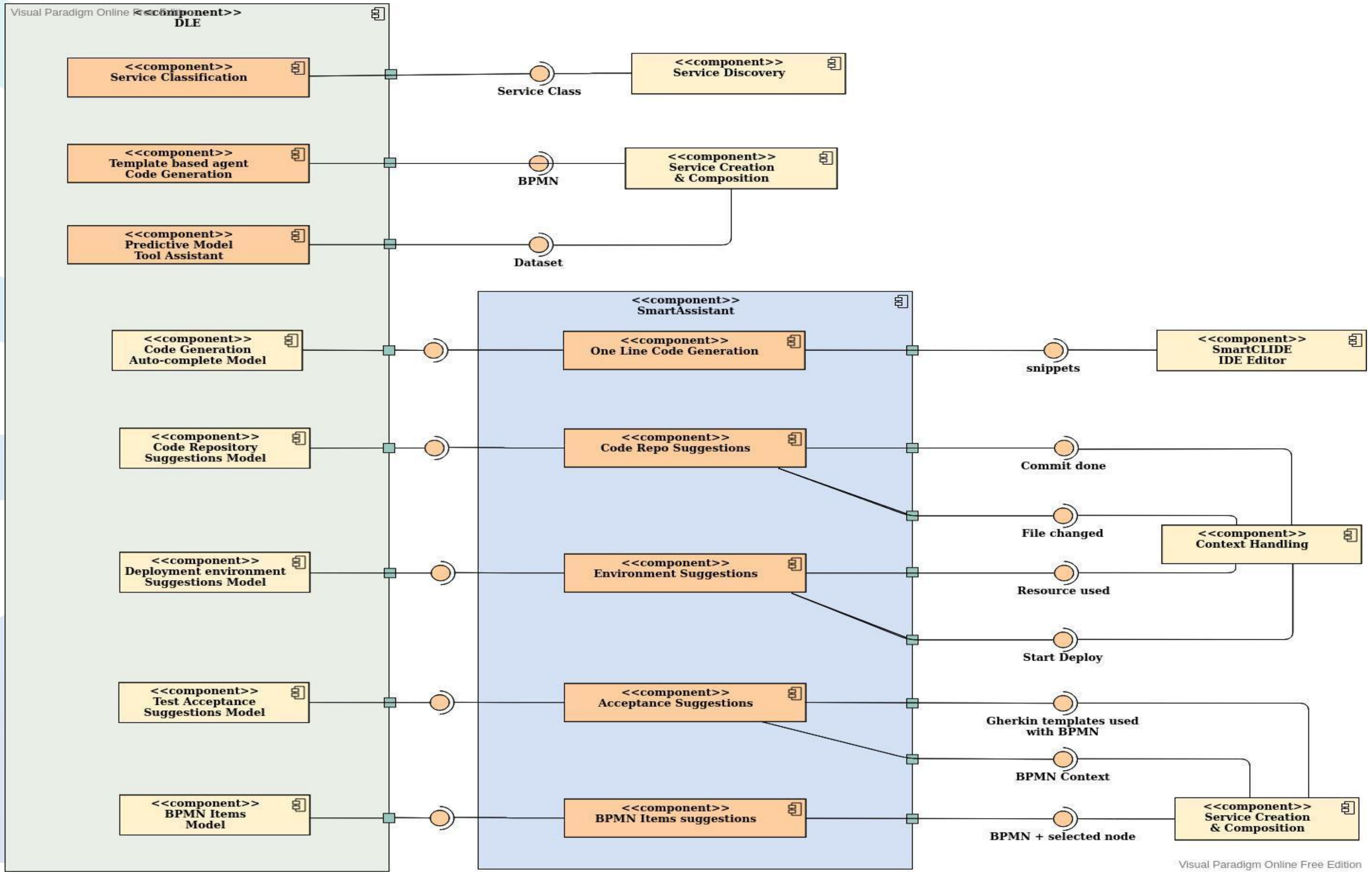


Gitlab	Dockerhub	Bitbucket	LIB Programmable	Programmable web	Common Attribute
full_name, name_with_namespace	name	full_name	SDK Name	Framework Name	Service Name
description	description	description	Description	description	Description
path path_with_namespace default_branch web_url	-	link	URL	Address URL	Service URL
ssh_url_to_repo http_url_to_repo web_url	github	link_github	Repository	Repository	Repository URL
language	-	language	language	language	Language
tag_list, keyword	keyword	type keyword	Category	Category	Class/Category/ Label
avatar_url forks_count star_count last_activity_at	stars automated? official?		Related APIs	Published Meta_Url	None-functional Attribute (Score)

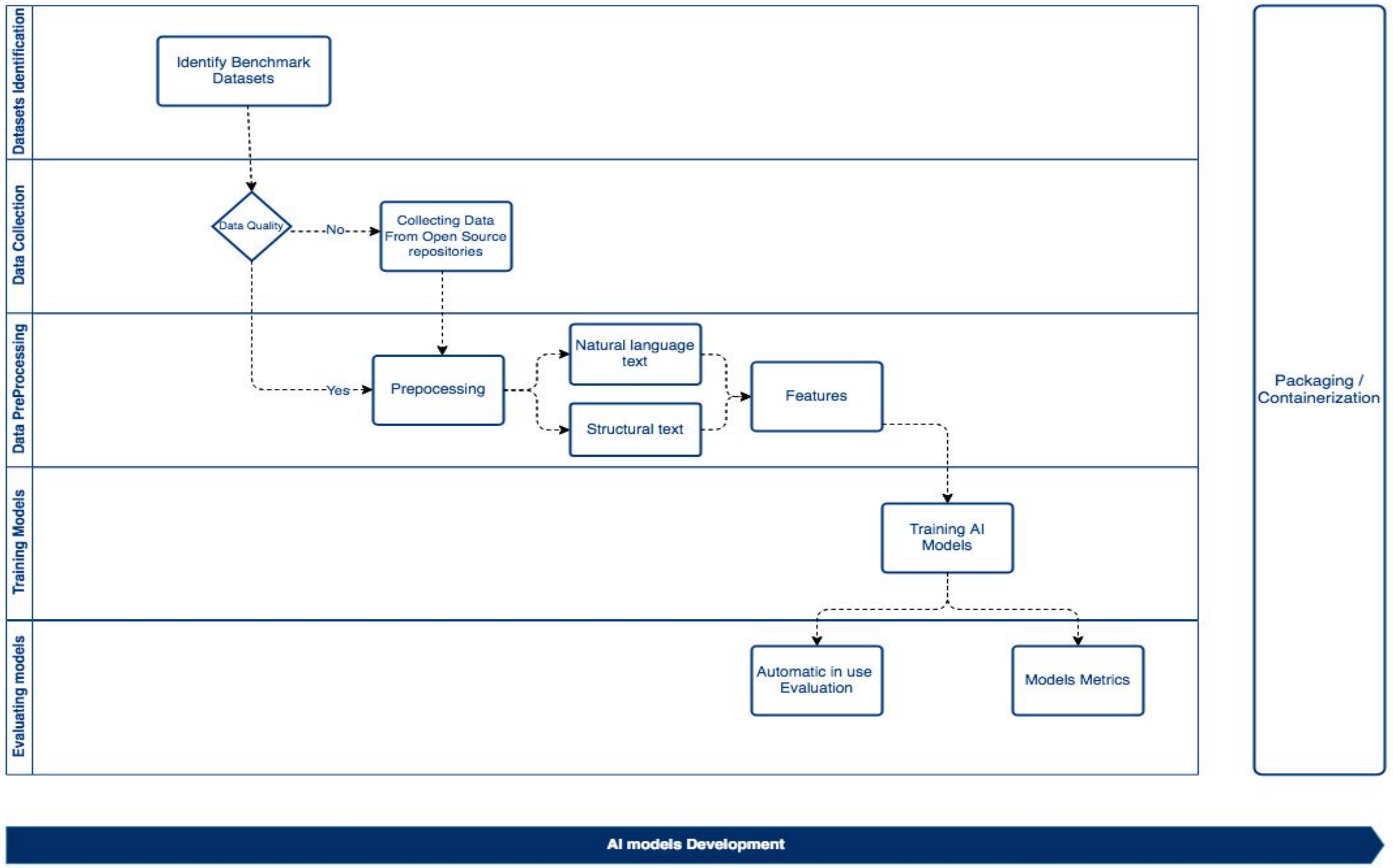
Programmableweb Services Types



DLE Component Diagram

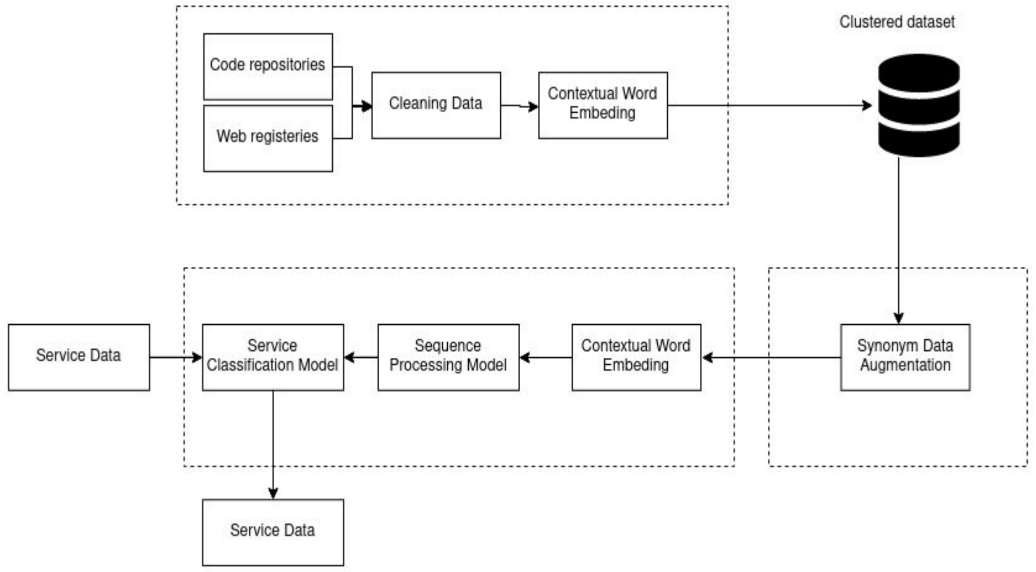

Visual Paradigm Online Free Edition
<<component>> DLE
<<component>> Service Classification
<<component>> Template based agent Code Generation
<<component>> Predictive Model Tool Assistant
<<component>> Code Generation Auto-complete Model
<<component>> Code Repository Suggestions Model
<<component>> Deployment environment Suggestions Model
<<component>> Test Acceptance Suggestions Model
<<component>> BPMN Items Model
Service Class
BPMN
Dataset
<<component>> Service Discovery
<<component>> Service Creation & Composition
<<component>> SmartAssistant
<<component>> One Line Code Generation
<<component>> Code Repo Suggestions
<<component>> Environment Suggestions
<<component>> Acceptance Suggestions
<<component>> BPMN Items suggestions
snippets
<<component>> SmartCLIDE IDE Editor
Commit done
File changed
Resource used
Start Deploy
<<component>> Context Handling
Gherkin templates used with BPMN
BPMN Context
BPMN + selected node
<<component>> Service Creation & Composition
Visual Paradigm Online Free Edition

DLE Models Development

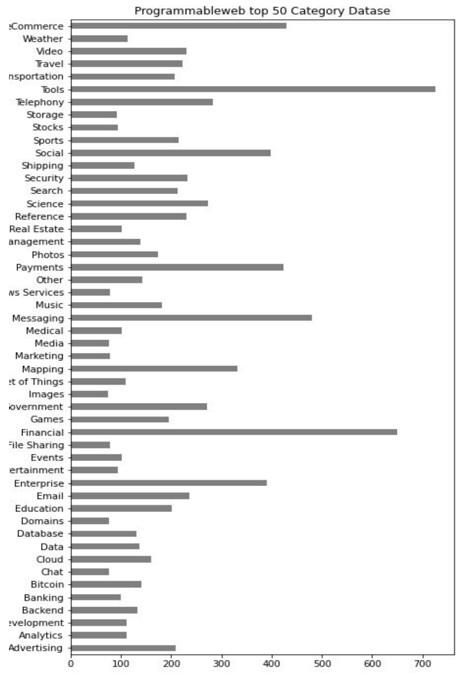


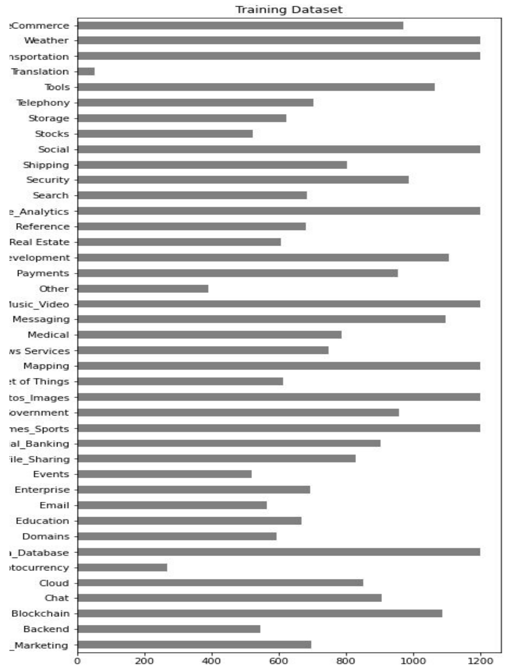
Example Model:Service Classification

Service Classification Architecture



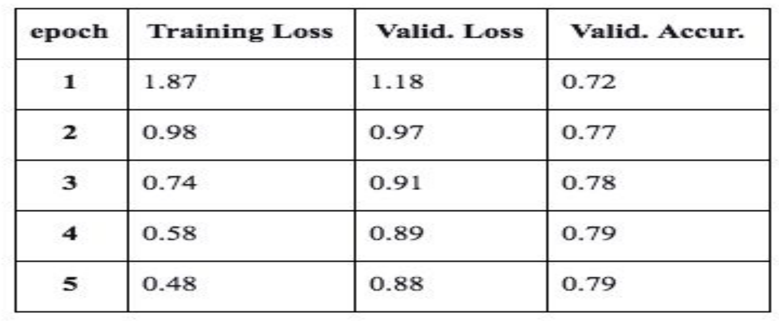
Benchmark Dataset vs Training Dataset





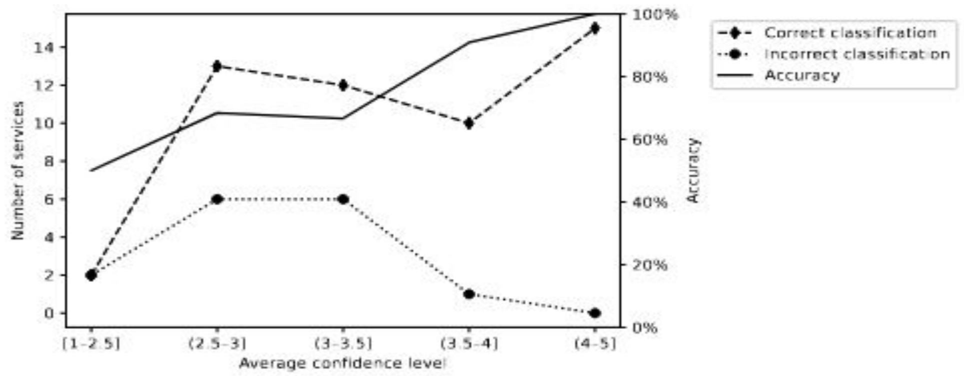
Service Classification Validation

Model Metrics

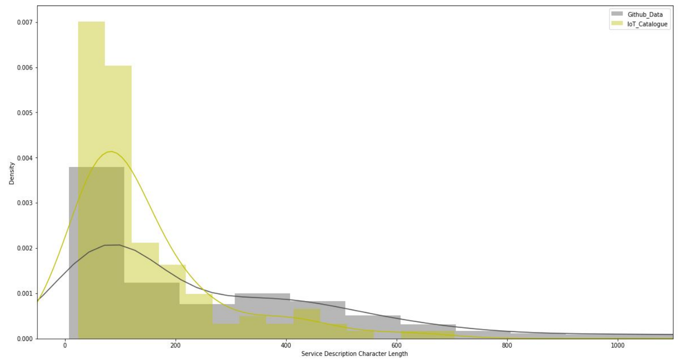


epoch	Training Loss	Valid. Loss	Valid. Accur.
1	1.87	1.18	0.72
2	0.98	0.97	0.77
3	0.74	0.91	0.78
4	0.58	0.89	0.79
5	0.48	0.88	0.79

Approach Performance vs. Experts Confidence

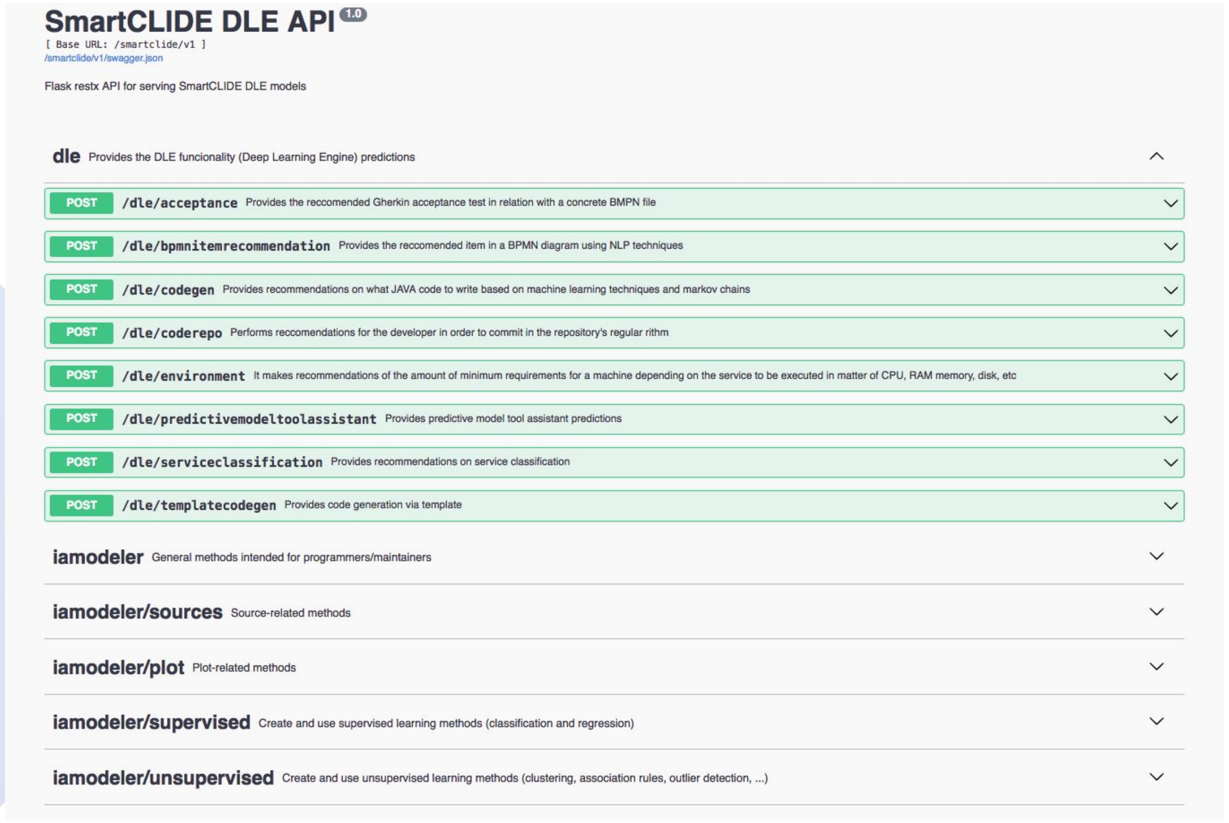


Training Service Desc vs Real-world data



Serving AI Models Via REST API

- Prediction
- Recommendation
- AutoML APIs



SmartCLIDE DLE API 1.0

[Base URL: /smartclide/v1]

/smartclide/v1/swagger.json

Flask restx API for serving SmartCLIDE DLE models

dle Provides the DLE funcionality (Deep Learning Engine) predictions

- POST `/dle/acceptance` Provides the reccomended Gherkin acceptance test in relation with a concrete BMPN file
- POST `/dle/bpmnitemrecommendation` Provides the reccomended item in a BPMN diagram using NLP techniques
- POST `/dle/codegen` Provides recommendations on what JAVA code to write based on machine learning techniques and markov chains
- POST `/dle/coderepo` Performs reccomendations for the developer in order to commit in the repository's regular rithm
- POST `/dle/environment` It makes recommendations of the amount of minimum requirements for a machine depending on the service to be executed in matter of CPU, RAM memory, disk, etc
- POST `/dle/predictivemodeltoolassistant` Provides predictive model tool assistant predictions
- POST `/dle/serviceclassification` Provides recommendations on service classification
- POST `/dle/templatecodegen` Provides code generation via template

iamodeler General methods intended for programmers/maintainers

iamodeler/sources Source-related methods

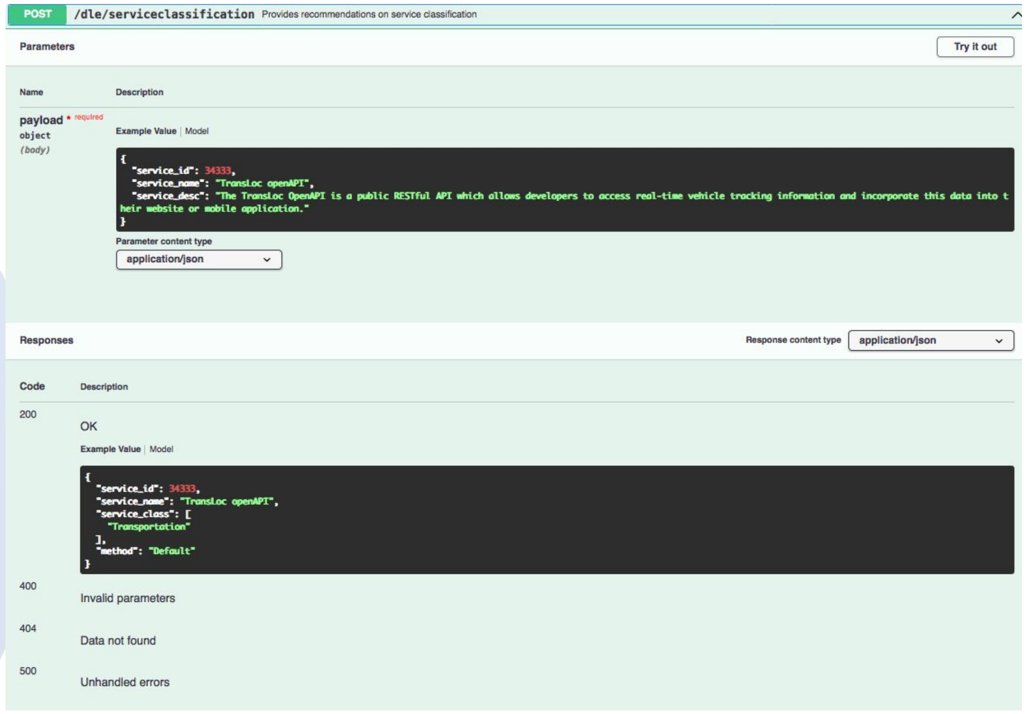
iamodeler/plot Plot-related methods

iamodeler/supervised Create and use supervised learning methods (classification and regression)

iamodeler/unsupervised Create and use unsupervised learning methods (clustering, association rules, outlier detection, ...)

Serving AI Models Via REST API

- Service Classification
- BPMN Items suggestions
- Acceptance test suggestions
- Deployment environment suggestions



/dle/serviceclassification Provides recommendations on service classification

Name	Description
payload * required object (body)	Example Value | Model

```
{
  "service_id": 34333,
  "service_name": "TransLoc openAPI",
  "service_desc": "The TransLoc OpenAPI is a public RESTful API which allows developers to access real-time vehicle tracking information and incorporate this data into t
heir website or mobile application."
}
```

Parameter content type: application/json

Responses — Response content type: application/json

Code	Description
200	OK Example Value | Model

```
{
  "service_id": 34333,
  "service_name": "TransLoc openAPI",
  "service_class": [
    "Transportation"
  ],
  "method": "Default"
}
```

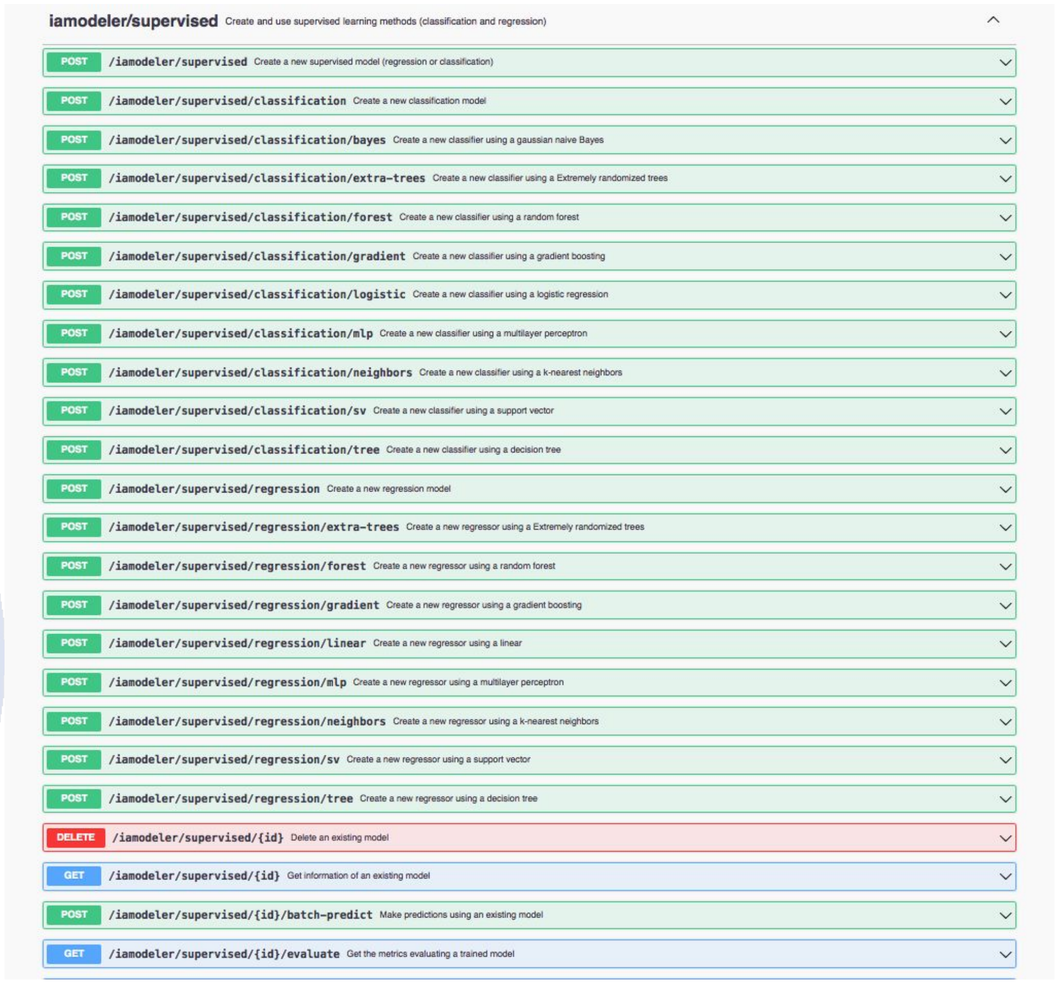
Code	Description
400	Invalid parameters
404	Data not found
500	Unhandled errors

Serving AI Models Via REST API

AutoML APIs

This subcomponent utilized the automated machine learning (AutoML) concept, allowing users to define ML actions sequences via an interface. These sequences contain the Predictive model tool APIs, which include 4 primary steps.

1. Importing data
2. Creating a supervised model based on regression or classification Model
3. Performing Prediction based on user input
4. Providing validation matric results that can use for visualization



iamodeler/supervised Create and use supervised learning methods (classification and regression)

POST	`/iamodeler/supervised`	Create a new supervised model (regression or classification)
POST	`/iamodeler/supervised/classification`	Create a new classification model
POST	`/iamodeler/supervised/classification/bayes`	Create a new classifier using a gaussian naive Bayes
POST	`/iamodeler/supervised/classification/extra-trees`	Create a new classifier using a Extremely randomized trees
POST	`/iamodeler/supervised/classification/forest`	Create a new classifier using a random forest
POST	`/iamodeler/supervised/classification/gradient`	Create a new classifier using a gradient boosting
POST	`/iamodeler/supervised/classification/logistic`	Create a new classifier using a logistic regression
POST	`/iamodeler/supervised/classification/mlp`	Create a new classifier using a multilayer perceptron
POST	`/iamodeler/supervised/classification/neighbors`	Create a new classifier using a k-nearest neighbors
POST	`/iamodeler/supervised/classification/sv`	Create a new classifier using a support vector
POST	`/iamodeler/supervised/classification/tree`	Create a new classifier using a decision tree
POST	`/iamodeler/supervised/regression`	Create a new regression model
POST	`/iamodeler/supervised/regression/extra-trees`	Create a new regressor using a Extremely randomized trees
POST	`/iamodeler/supervised/regression/forest`	Create a new regressor using a random forest
POST	`/iamodeler/supervised/regression/gradient`	Create a new regressor using a gradient boosting
POST	`/iamodeler/supervised/regression/linear`	Create a new regressor using a linear
POST	`/iamodeler/supervised/regression/mlp`	Create a new regressor using a multilayer perceptron
POST	`/iamodeler/supervised/regression/neighbors`	Create a new regressor using a k-nearest neighbors
POST	`/iamodeler/supervised/regression/sv`	Create a new regressor using a support vector
POST	`/iamodeler/supervised/regression/tree`	Create a new regressor using a decision tree
DELETE	`/iamodeler/supervised/{id}`	Delete an existing model
GET	`/iamodeler/supervised/{id}`	Get information of an existing model
POST	`/iamodeler/supervised/{id}/batch-predict`	Make predictions using an existing model
GET	`/iamodeler/supervised/{id}/evaluate`	Get the metrics evaluating a trained model

SmartCLIDE

Thank you!