High-Level Design

Model Evaluation and diagnosis display

Revision Number: 1.0
Last date of revision: 03/07/21

Contents

1. Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) document is to add the necessary and important details to the current project description to represent a suitable model for the solution we provide to the existing problem. The high-level design was selected by deciding what aspects of the current solution were most important and then building architecture around them. This document can be used as a reference manual for how the modules interact at a high level.

1.2 Scope

This documentation depicts the structure of our solution such as the database architecture, application flow, and technology architecture. It uses non-technical to mildly-technical terms which should be easily understandable by the end-users as well as the administrators of the system.

1.3 Overview

The HLD will

- Present all of the design aspects and define them in detail
- Include design features and the architecture of the project
- Describe the user interface that is being implemented
- Describe the hardware and software interfaces
- list and describe the non-functional attributes like portability, reliability, maintainability, etc

General Description:

2.1 Product perspective

This product is supposed to solve the problem that most data scientists come across concerning which machine model works best for a particular dataset or which model should proceed over to production if they have a bunch of them and they need to select one. The product focuses on providing evaluation reports for a machine learning model to let data scientists know what is the performance of their model on unseen data. The solution we provide has a context-based visualization capability as well.

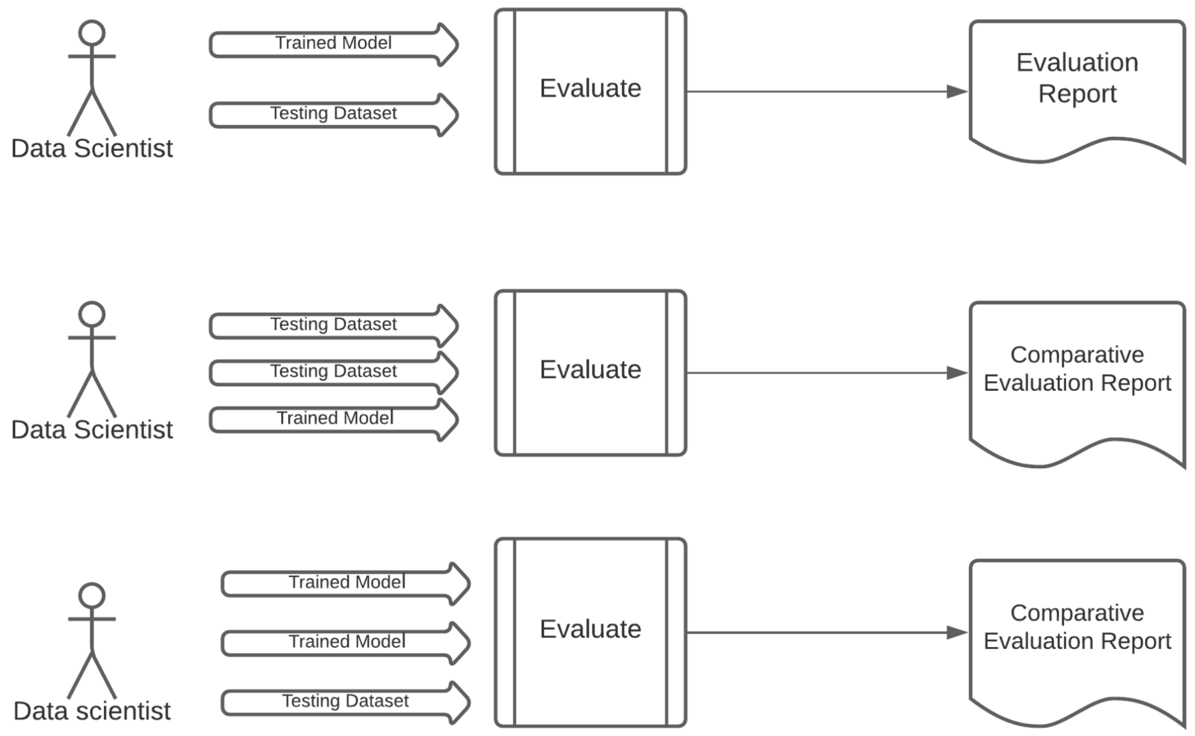
2.2 Requirements

The main goal of the product is focused on evaluating the performance of several machine learning models across different validation datasets. It also provides the capability to compare the performance of multiple models on the basis of different metrics. However, comparison of model evaluation is only possible for the following 2 scenarios

- Evaluation reports for one model against different validation datasets(having the same schema)
- Evaluation report for multiple models (2 or more) generated against same validation dataset

Apart from the comparison, the user should be able to visualize the performance report for a single model evaluation. He/She should also be able to add user-defined metrics and it's value for this particular use case corresponding to a specific evaluation ID. The user should be able to persist the newly added evaluation metrics.

2.3 High-level architecture



2.4 Tools Used

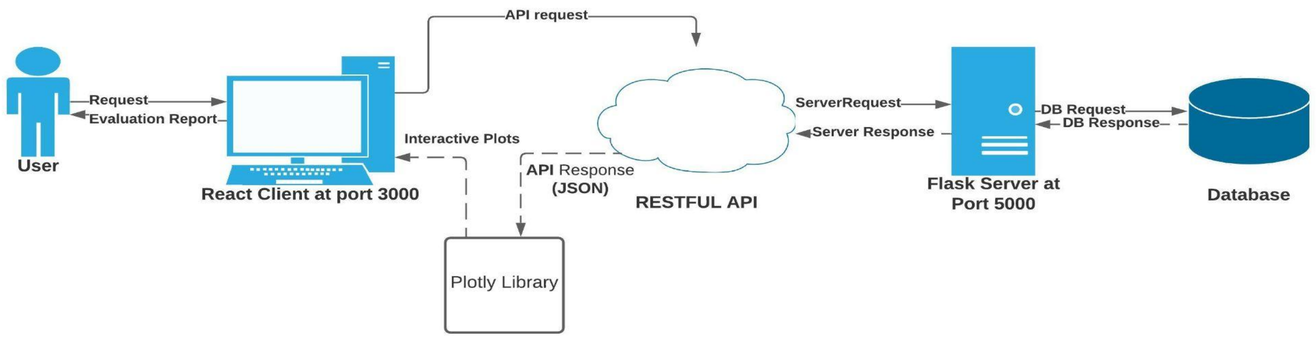
1. **Lucid chart**, a web-based proprietary platform that allows users to collaborate on drawing, revising, etc, and **plantuml**, an open-source tool allowing users to create diagrams from a plain text language is used to generate all of the diagrams used in the analysis and design phases of the project.
2. **Flask,** a micro web framework written in Python has been used to serve the backend for this project.
3. **SQLAlchemy** a Python toolkit has been used for an object-relational mapper (ORM). It provides a set of tools that lets us interact with our database models consistently across database engines.
4. **SQLite** has been used as a database storage engine.
5. **React JS**, a free and open-source front-end JavaScript library has been used for building user interfaces or UI components.
6. **Plotly-React** has been used for plotting interactive charts and graphs.
7. **Swagger-UI** to trigger REST endpoints or visualize them.

3. Design Details

3.1. Main Design Features

The main design features include five major parts: the architecture, the user input design, the user interface design, REST APIs, the database, and process relation. In order to make these designs easier to understand, the design has been illustrated in the attached diagrams (ER, Use Case, and Screenshots).

3.2 Application Architecture



3.3 Technology Architecture

The front end of the program is a React-based web application. The functionality will vary based on user requirements whether a user wants a single model evaluation report or comparative evaluation report. There is no login functionality as this was not a part of the requirement of this product but can be added later.

3.4 Standards

- Database – Relational
- Inputs – User provides information through Model Training and Dataset creation which is later stored in a database.
- Security – No such login aspect as of now but can be added.
- Quality – By keeping the interface simple and direct, quality should be kept at a maximum

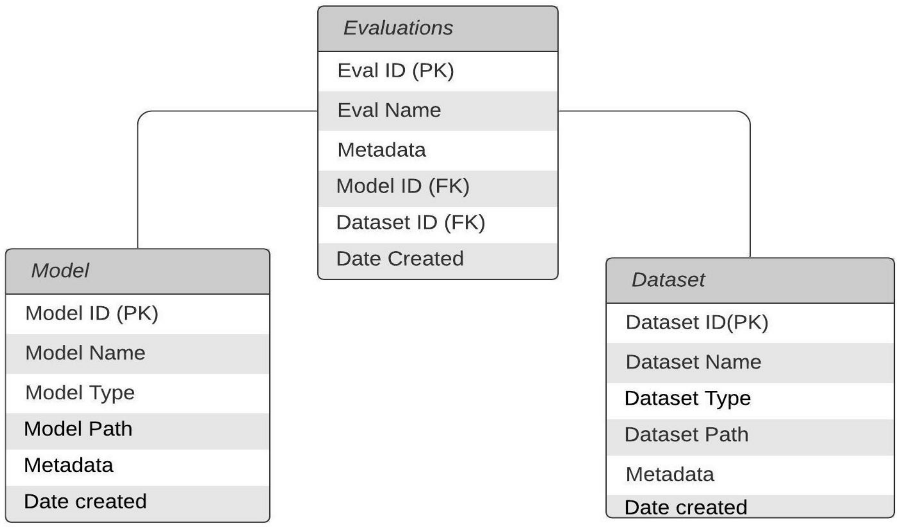
3.5 Database Design

When it comes to choosing a database for this product, the biggest decision is picking a **relational (SQL)** or **non-relational (NoSQL)** data structure. While both the databases are viable options still there are certain key differences between the two that administrators must keep in mind when making a decision.

NoSQL Database	**Relational Database**
They have dynamic schema	These databases have fixed or static or predefined schema
These databases are best suited for hierarchical data storage.	These databases are not suited for hierarchical data storage.
These databases are not so good for complex queries	These databases are best suited for complex queries
Horizontally scalable	Vertically Scalable
Follows CAP(consistency, availability, partition tolerance)	Follows ACID property

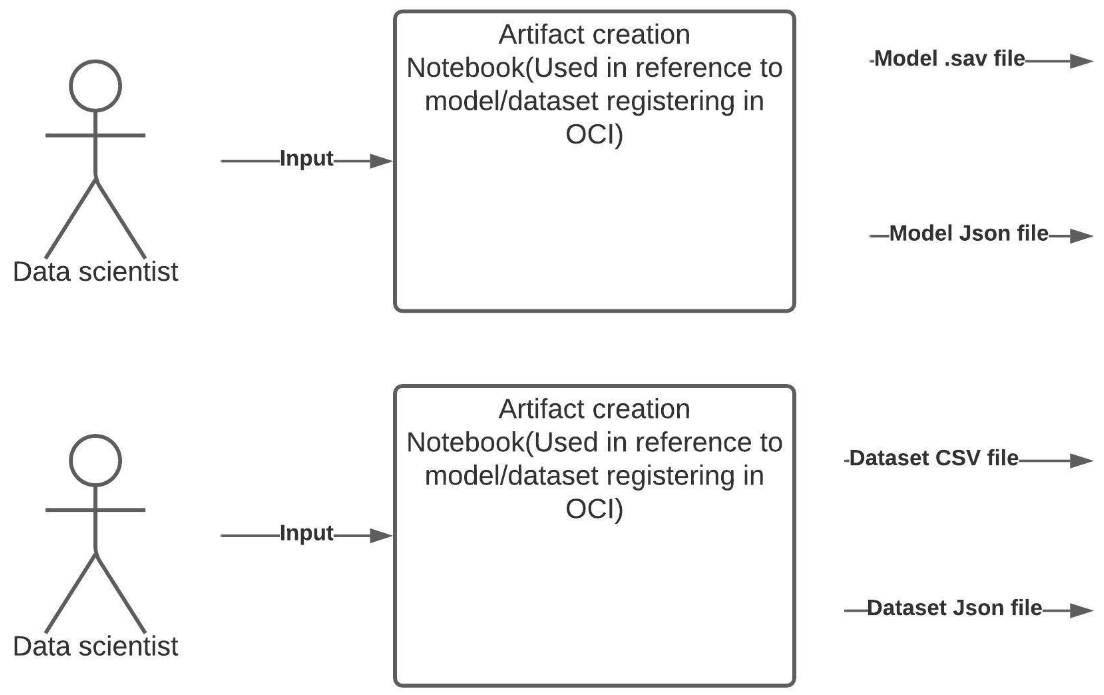
We chose RDBMS for our product because there was no horizontal scaling requirement and we had fixed schemas for our data points.

Below is the schema of our application DB Design:



3.6 Inputs

Model and Dataset Registration


Data scientist
Input
Artifact creation Notebook(Used in reference to model/dataset registering in OCI)
Model .sav file
Model Json file
Data scientist
Input
Artifact creation Notebook(Used in reference to model/dataset registering in OCI)
Dataset CSV file
Dataset Json file

The content of the JSON file of dataset will be

```
dataset_metadata = {
    "author":Name of the person who’s registering the dataset,
    "label":target column name,
    "copy":dataframe._is_copy,
    "dataset_split_method": ex- train_test_split,
    "use_case_type":[classification/regression/clustering]
}
```

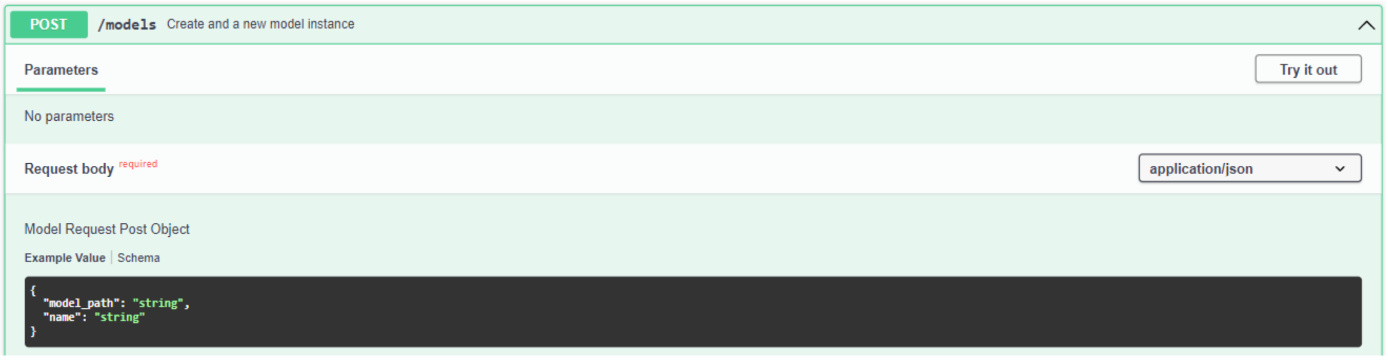
The content of the JSON file of model will be

```
model_metadata = {
    "use_case_type":"",
    "library": sklearn.<whatever library you are using>,
    "model":" ",
    "algorithm": " ",
    "hyperparameters":{

     },
    "library_version": str(sklearn.__version__),
    "author":""
}
```

Once the user provides the necessary information for registering models and datasets, they can be stored in a database using REST endpoints of models and datasets resources.

The following is the JSON payload of the POST method of model Resource



No parameters

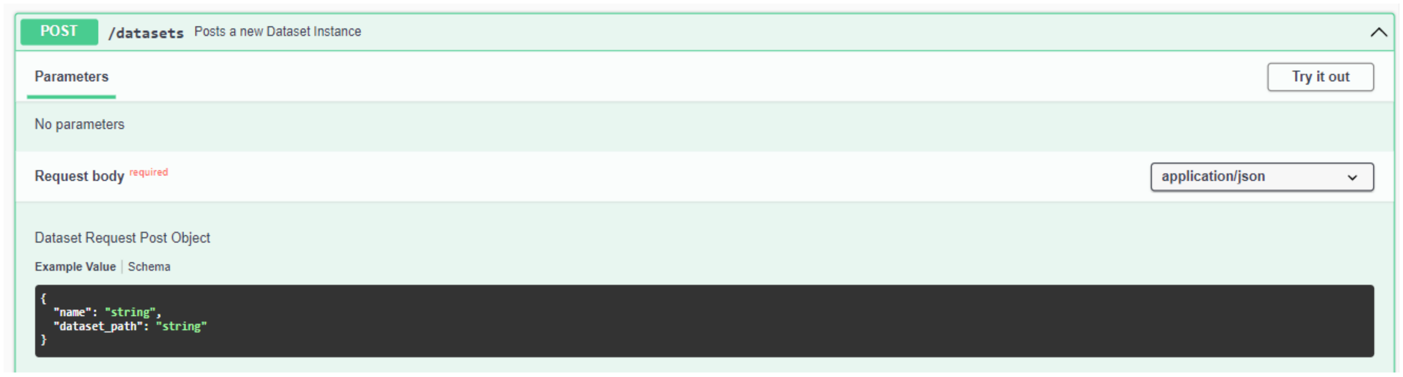
Request body required application/json

Model Request Post Object

Example Value | Schema

```
{
  "model_path": "string",
  "name": "string"
}
```

The following is the JSON payload of the POST method of dataset Resource



POST /datasets Posts a new Dataset Instance

Parameters

No parameters

Request body required

Dataset Request Post Object

```
{
  "name": "string",
  "dataset_path": "string"
}
```

3.6 Hardware and platforms requirements

Processor

Processor performance depends not only on the clock frequency of the processor but also on the number of processor cores and the size of the processor cache.
The following are the processor requirements for this product:
Minimum:

- 1.4 GHz 64-bit processor
- Compatible with x64 instruction set

RAM

The following are the estimated RAM requirements for this product:
Minimum: 512 MB (2 GB for Server with Desktop Experience installation option)

Storage controller and disk space requirements

The following are the estimated minimum disk space requirements for the system partition.
Minimum: 32 GB

Operating System

Since the application is dockerized, it can run on any compatible OS

Packages Installed

- Flask==2.0.1
- Flask-RESTful==0.3.9
- Flask-SQLAlchemy==2.5.1
- gunicorn==20.1.0
- Jinja2==3.0.1
- joblib==1.0.1
- kiwisolver==1.3.1
- numpy==1.20.3
- pandas==1.2.4
- pyparsing==2.4.7
- python-dateutil==2.8.1
- scikit-learn==0.24.2
- scipy==1.6.3
- sklearn==0.0
- threadpoolctl==2.1.0
- Werkzeug==2.0.1
- flask-swagger-ui

3.7 Files

This product will not use a large number of files. The API endpoints can be tweaked easily in ***api/app.py***. The front-end logic for consuming our API is contained in ***client/src/***. The code contained within these files simply exists to demonstrate how our front-end might consume our back-end API.

```
Root Folder/
|
+---api/
|    |   Dataset/
|    |   Model/
|    |   models/
|    |   Resources/
|    |
|    +---static/
|    |          swagger.json
|    |   app.py
|    |   api.flaskenv
|    |   data.db
|    |   db.py
|    |   Dockerfile
|    |   Requirements.txt
|
|
+---client
|    |  public/
|    |  src/
|    |  Dockerfile
|    |  package.json
|    |  yarn.lock
|
+--- docker-compose.yml
```

3.8 Evaluation Reports

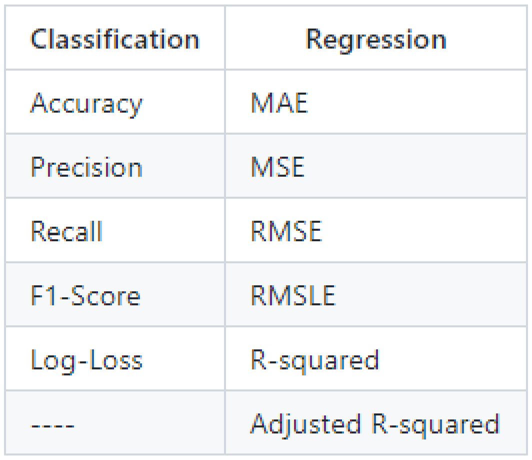
3.8.1 Single Model Evaluation

The evaluation metrics for a single model can be visualized by clicking on the button encircling the Evaluation ID of the evaluation in the table at the Homepage. It essentially sends a get request for the evaluation and based on the received payload, It will render the visualizations as follows:

Note: All tables rendered in this scenario are semi-interactive, except the table for feature importance.

Evaluation Metrics

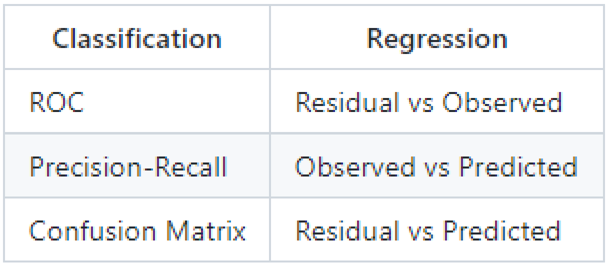
Following evaluation metrics will be visible to the user in a tabular, bar chart, and line chart format:



Classification	Regression
Accuracy	MAE
Precision	MSE
Recall	RMSE
F1-Score	RMSLE
Log-Loss	R-squared
----	Adjusted R-squared

Curves and Charts

The following curves and charts will be shown to the user when they select this option:



Classification	Regression
ROC	Residual vs Observed
Precision-Recall	Observed vs Predicted
Confusion Matrix	Residual vs Predicted

Along with the plots, there are also several ways to interact with them, by:

1. Zooming in and out using scrolling
2. Select
3. Lasso select
4. The slider of the cutoff value
5. Button to reset to initial state
6. Axis zoom in and out
7. Drag and move
8. Save as PNG

Dataset Information

The following data is shown about the test dataset used by the model for the prediction:

1. Dataset Statistics are shown in tabular format as well as in Line Chart format. Following are the statistics displayed in it, for each column of the dataset:
    - Mean
    - Standard deviation
    - Minimum value
    - Maximum value
    - First Quartile
    - Second Quartile
    - Third Quartile
    - IQR
    - Number of missing values
2. Feature Importances are shown in Bar chart as well as tabular format
3. Class Imbalance is shown in Pie-chart format

Model Information

This section gives a tabular view of the parameters and attributes that are associated with the trained model, in a tabular format.

Details

Each of the above tabs will have a Details tab, that gives information about the evaluation in general, some information about the dataset, and the model used in the evaluation.

3.8.2 Multiple Model Single Dataset Comparison

For both regression and classification, there are five types of components rendered:

- Metrics
- Dataset Information
- Model information
- Curves
- Details(part of each tab panel)

Metrics

A semi-interactive table, along with both bar graphs and line charts are rendered in this tab. Metrics are the same as put up in the above section on Single Evaluation. The table can be sorted by metrics to compare the models.

Dataset information

Since the evaluations being considered in this case must have the same dataset, the same component that was used for a single evaluation use case has been used.

Model Information

Multiple tables listing out parameters and attributes of each model are rendered.

Curves

The plots mentioned in the above section are rendered, with the traces of other models in the same graph.

Details

Every tab panel has it. It's the same as single evaluation, except now, it has tabs for all evaluations that were selected by the user.

3.8.3 Single Model Multiple Datasets

For both regression and classification, there are five types of components rendered:

- Metrics
- Dataset Information
- Model information
- Curves
- Details(part of each tab panel)

Metrics

A semi-interactive table, along with both bar graph and line charts are rendered in this tab. Metrics are the same as put up in the above section on Single Evaluation. The table can be sorted by metrics to compare the datasets.

Dataset information

The dataset statistics for all datasets are shown tab-wise. Users can switch between statistics and compare the datasets based on those statistics. The user can also switch between tabular view and Line Chart view. Along with this, it also contains information about the feature importances of the datasets in the chart and tabular format and the class imbalance.

Model Information

Since the model being used is the same, there is a single table listing out all the parameters and attributes of the trained model.

Curves

The plots mentioned in the 'Single Model Evaluation' section are rendered, with the traces of other datasets in the same graph, or in subplots.

Details

Every tab panel has it. It’s the same as a single evaluation, except now, it has tabs for all evaluations that were selected by the user.

3.9 User Interface

Homepage

The Homepage consists of an interactive table where all the evaluations are listed. By clicking on the Evaluation ID, the user can see the visualizations of the performance of his/her model. The Compare button can be used to compare two or more evaluations. Clicking on the Evaluation ID triggers the model evaluation if the metrics are not there already.

Each row has the following information:

- Evaluation ID
- Evaluation name
- Model Type
- Model
- Dataset
- Date Created

Evaluation Post Form

This page contains the form that helps the user to register for an evaluation. The user enters the following information here:

- Evaluation name(Text field)
- Dataset(selection)
- Model(selection)
- Description(optional)

3.10 Portability

This system should have the ability that, once it is together, the entire system should be able to be physically moved to any location. Code and program portability should be possible between any OS. For everything to work properly, all components should be compiled from the source. Since the application is dockerized completely, portability is ensured.

3.11 Security

Because security is not the prime focus of this project, only the minimal aspects of security will be implemented.

3.12 Reusability

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

4. Limitations

- Currently, the product supports evaluation reports for models trained using the sklearn library only but is extendible to other libraries as well.
- The use-cases supported as of now are Regression and classification but are extendible to support clustering as well.
- Model files are unpickled and used. So only one extension, .sav, is supported