Architecture

Credit Card Default Prediction

Revision Number: 1.0

Last date of revision: 27/09/2023

# 

# 

# 

# 

# 

# 

# 

# 

# Document Version Control

**Change Record**

| Date Issued | Version | Description | Author |
| --- | --- | --- | --- |
| 25/09/2023 | v1.0 | First Draft | Abraham Audu |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[Document Version Control 2](#_heading=h.gjdgxs)

[Abstract 4](#_heading=)

[1. **Introduction**](#_heading=) 5

[1.1 Why this Architecture Document? 5](#_heading=)

[1.2 Scope 5](#_heading=h.d1qeboog8101)

[1.3 Constraints 5](#_heading=h.25had5vi2514)

[1.4 Risks 5](#_heading=h.ko1w5tjrieq1)

[1.5 Out of Scope 5](#_heading=h.3mgtvva5v9t0)

[**2. Technical Specifications 6**](#_heading=h.ggkmsajlo83h)

[2.1 Dataset Overview 6](#_heading=h.ik6kx6qkz252)

[2.2 Predicting Monthly Payment Default 6](#_heading=h.3a339y863l5j)

[2.3 Logging 6](#_heading=h.7nfstg7w5zjh)

[2.4 Deployment 6](#_heading=h.u0x9ys6hgr05)

[**3. Technology Stack 7**](#_heading=h.6vm6ejoevwnp)

[**4. Proposed Solution 7**](#_heading=h.anf9p88rjr2c)

[**5. Model Training and Validation Workflow 7**](#_heading=h.ut8anbobgfdx)

[**6. User I/O Workflow 8**](#_heading=h.1l7gex6ibgcx)

[**7. Exceptional Scenarios 8**](#_heading=h.l3ugmcy4seyh)

[**8. Unit Test Cases 9**](#_heading=h.m04g3yo1j0pj)

[**9. Key Performance Indicators (KPIs) 9**](#_heading=h.36tw07kh7b0k)

# Abstract

Banking institutions offer credit to customers as one of their major businesses and this comes with some associated credit risk. WIth millions of customers, it is important to have an automated system which can filter through customer information and financial history to determine the probability that a card owner will default in order to limit their access to current or future credit services and minimise risk to the issuing organisation. This project covers the implementation of a machine learning based platform which is able to take a credit card owner’s information and financial history and predict the probability that they will be credit defaulters.

# Introduction

## Why this Architecture Document?

The purpose of this architecture document is to present a detailed description of the Credit Card Default Prediction solution. It will explain the purpose and features of the system, the interfaces of the system, how the system will operate and the reaction of the system to specific action by the end user. This document is intended for both stakeholders and the developers of the system, and will be proposed to higher management for its approval.

The main objective of the project is to predict if a credit card owner will default on future credit card payments based on financial records. These financial records consist of basic demographics such as age, level of education, marital status, etc., as well as financial information such as total credit card dates, credit card bills for a given time period and bill payments, including level of punctuality in payment.

This project shall be delivered in two phases:

Phase 1: Model development

Phase 2: API and UI development

## 1.2 Scope

This software system will be a web application designed to predict the probability tht a credit card owner will default on their payments so as to help the card issuing institution make informed decisions based on the feedback of the system.

## 1.3 Constraints

The system will only work with the dataset structure of a particular card issuing organisation and will need to be retrained to work with different structures of financial information based on specifications provided.

## 1.4 Risks

As with all technological implementations, there is bound to be inaccuracies and glitches in the system. Seeing as this system is designed for a financial use case which can be sensitive, the tool should be used only as a guide and results which seem inaccurate should be rechecked by a professional with domain expertise.

## 1.5 Out of Scope

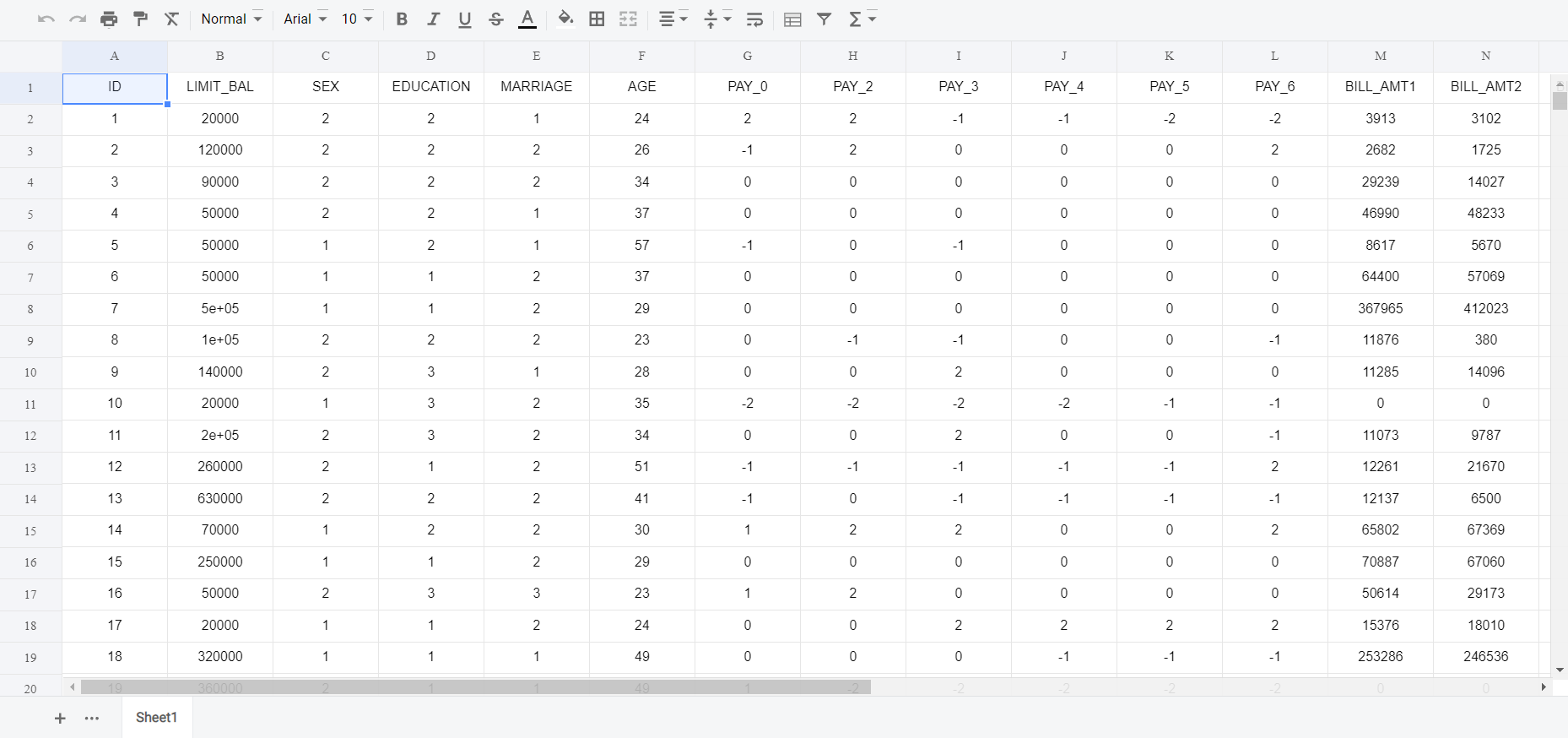
This project is designed to be a stand-alone software and does not involve integration into a full on financial suite of software.

# 2. Technical Specifications

## 

## 2.1 Dataset Overview

The data is a CSV file containing 30,000 rows of data with each row representing a customer. The data consists of personal information like age, gender, level of education and marital status. In addition the data has financial information covering punctuality of payments, credit card bills and corresponding credit card bill payments for the last six months.



## 2.2 Predicting Monthly Payment Default

* The user is prompted to upload credit card user data as CSV
* The system validates the data uploaded
* The user is prompted to choose the prediction model they want to use
* The system will then predict the probability that the credit card owner will default on their monthly payment

## 2.3 Logging

The system will log activities on three layers:

* The development pipeline for debugging purposes
* The server side
* User activity on the interface

Logs will be saved to dedicated files for the different layers of the application

## 2.4 Deployment

The Application will be deployed locally.

### 2.4.1 Local Deployment Details

Code Repository: <https://github.com/theabrahamaudu/credit_card_default_predictor>

Steps:

* Clone repository
* Create virtual environment
* Open terminal
* Install requirements: python -m pip install -r requirements.txt
* Start FastAPI server: python -m src/main
* Start Streamlit frontend: streamlit run src/streamlit\_frontend.py

Refer to README.md on the code base for more details.

### 2.4.2 Cloud Deployment

Server deployed on Render: <https://credit-card-predictor-api.onrender.com>



Frontend deployed on Streamlit: <https://creditcarddefaultapp.streamlit.app/>



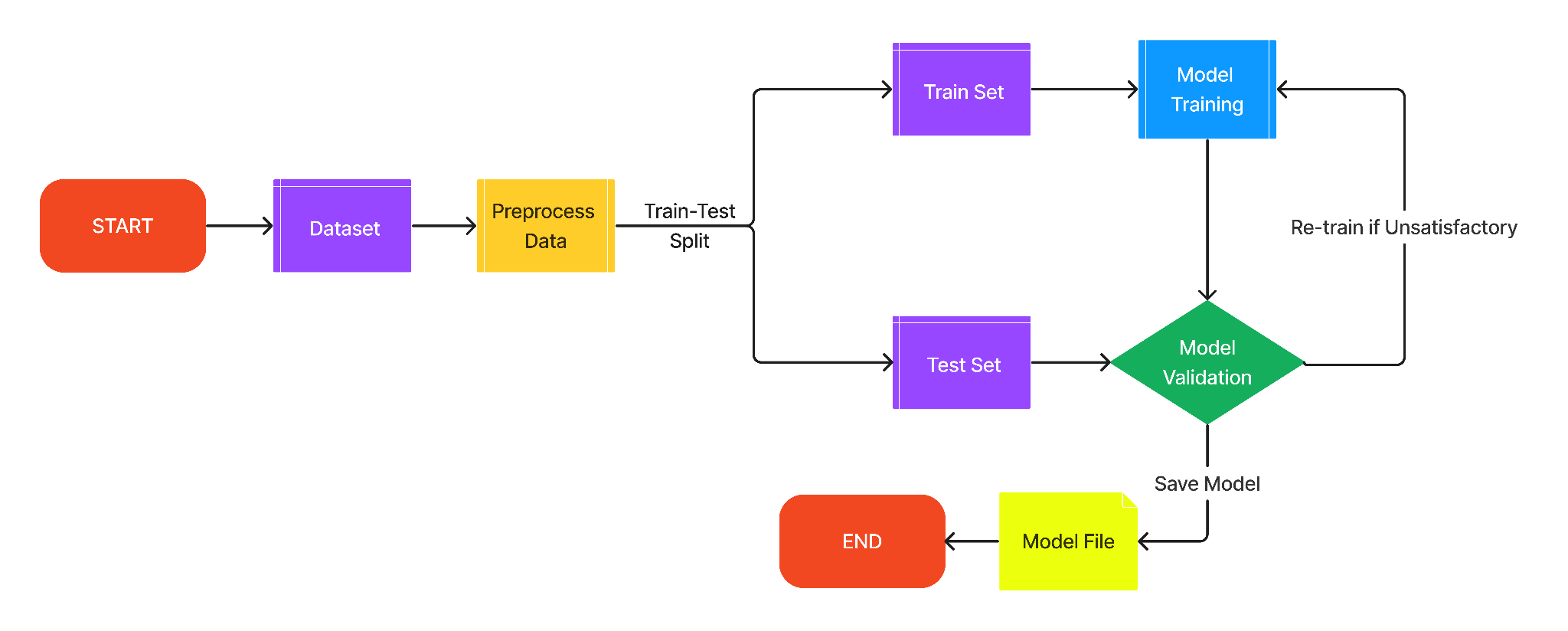
# 3. Technology Stack

| **Frontend** | Python Streamlit |
| --- | --- |
| **Backend** | Python FastAPI |
| **Deployment** | Local machine |

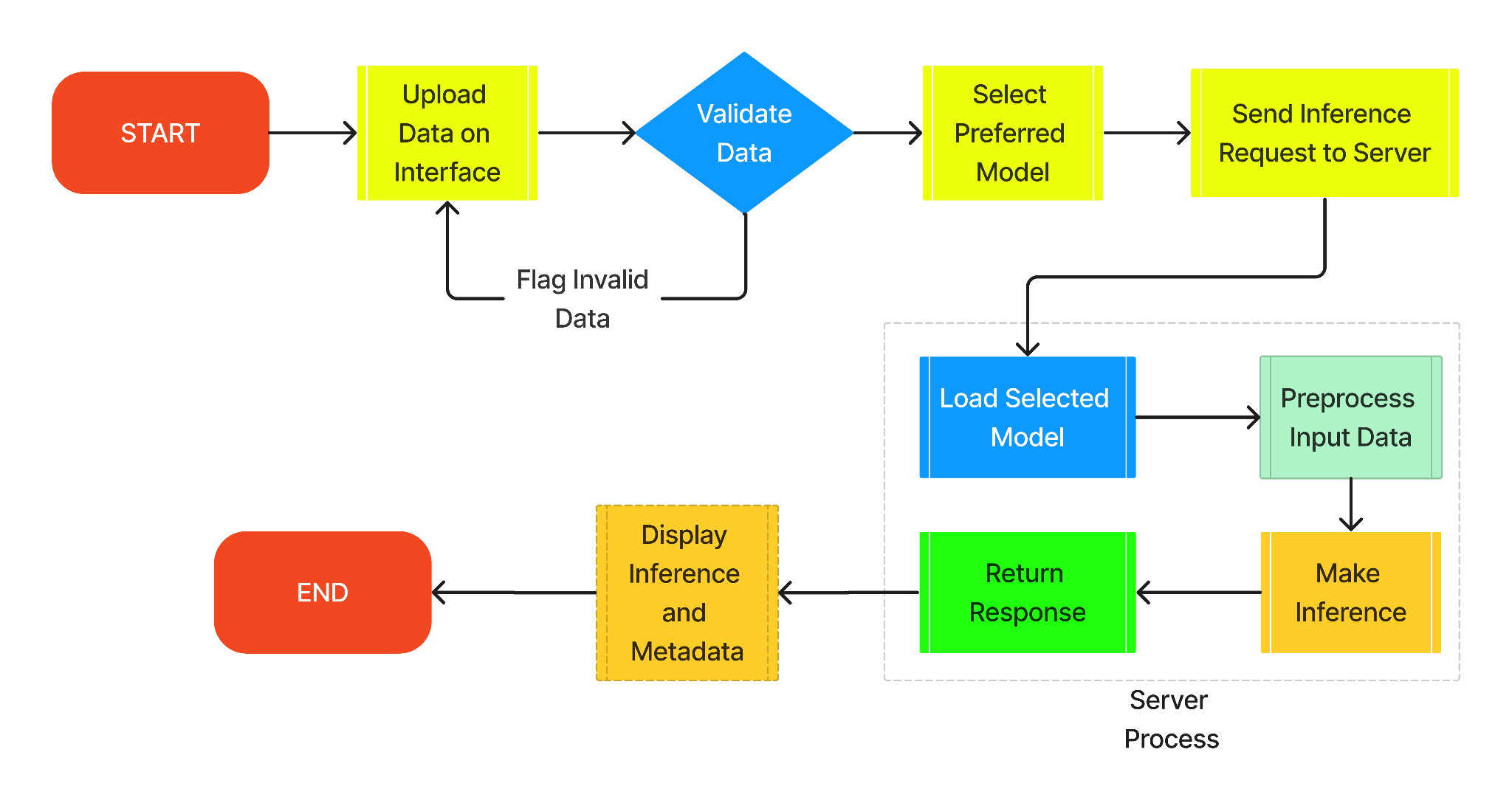
# 4. Proposed Solution

The problem is fundamentally a classification problem and so classification models will be considered. Models to be considered include Logistic Regression, Support Vector Machine, Multi-Layer Perceptron Classifier, Random Forest and Decision Tree. The models will be evaluated using weighted f1-scores to ensure they are able to predict both cases in a balanced manner.

# 5. Model Training and Validation Workflow



# 6. User I/O Workflow



# 7. Exceptional Scenarios

| **Step** | **Exception** | **Mitigation** | **Module** |
| --- | --- | --- | --- |
| Train data pre-processing | Error in any preprocessing step | Log the error to the pipeline log, raise exception | preprocess.py |
| Web UI data pre-processing | Error in any preprocessing step | Log the error to the pipeline log, raise exception | preprocess.py |
| User data upload | Error uploading file | Notifies the user of upload error | streamlit\_frontend.py |
| User data validation | Error validating data | Notify the user, log the error | streamlit\_frontend.py |
| Make API call to get prediction | Error in calling API or in response from API | Notify the user, log the error | streamlit\_frontend.py |
| Load selected model | Error loading trained model | Log the error | main.py |
| Pre-processing user data for inference | Error in the preprocessing script | Log error (preprocessing script logger gives details) | main.py |
| Inference and metadata | Error making inference | Log the error, send empty response | main.py |

# 8. Unit Test Cases

| **Test Case Description** | **Prerequisite** | **Expected Result** |
| --- | --- | --- |
| Verify whether undersampling function works with binary targets | target column is encoded | Oversampled targets are randomly removed |
| Verify whether undersampling function works with single value targets | target column is encoded | Number of target rows stays intact |
| Good data uploaded by user is positively validated | Data uploaded by user | validator passes |
| Data with unrecognised feature is uploaded by user | Data uploaded by user | Validator flags unrecognised feature |
| Data with incomplete set of features uploaded by user | Data uploaded by user | Validator flags unprovided feature as missing |
| Data with invalid data type uploaded by user | Data uploaded by user | Validator flags features with incorrect data type |

# 9. Key Performance Indicators (KPIs)

* Model accuracy
* Inference time
* Overall prediction time over the network