# Machine Learning Project Documentation

## 1. Deployment

## 1.1 Overview

In the deployment phase of a machine learning model, the trained model is made accessible for real-world applications, enabling user interaction and insights. This process begins with model training and evaluation, where the model is trained on historical data and its performance is assessed using relevant metrics. Once satisfied with the model's performance, it is serialized for reuse in the deployment environment, using libraries like TensorFlow. The deployment environment is then set up, which may involve configuring a server or cloud environment, and frameworks such as Gradio, Flask, or Fast API are used to serve the model. An API or user interface is built, often with Gradio, to allow users to input features and receive predictions in real-time. The deployment also involves scaling and monitoring to handle multiple users and large-scale data requests, ensuring the model's performance is tracked and maintained. Continuous improvement is achieved by using new data generated during production to periodically retrain or fine-tune the model, ensuring it adapts to changing trends and maintains accuracy.

## 1.2 Model Serialization

In the context of a TensorFlow/Keras project, serializing a model involves saving and loading the trained model for future use. The steps are as follows:

### 1.2.1 Saving the Model

To save a trained model, use the `model.save()` function. This function stores the model's architecture, learned weights, optimizer configuration, and the state of the model. The command to save the model is:

ann.save(“)

This command creates a file in the current directory, which contains all the necessary components to recreate and use the trained model later.

### 1.2.2 Loading the Model

To use the saved model in a production environment, reload it using the `load\_model` function from TensorFlow/Keras:

from tensorflow.keras.models import load\_model

ann = load\_model('model.h5')

This command loads the saved model into memory, making it ready for making predictions directly.

TensorFlow provides the SavedModel format, which is ideal for saving larger models, especially for production use. This format saves the model in a directory with a structured layout, making it suitable for deployment in various environments.

saved\_model. pb: A protocol buffer file that contains the model architecture.

variables: A directory that stores the model weights.

### 1.3 Model Serving

Gradio enables the rapid creation of an interface where users can input features and receive predictions from the model. It is ideal for testing and lightweight deployment purposes. Below are the steps to serve a model using Gradio:

1. Loading the Serialized Model:

Before serving the model, it must be deserialized back into memory to be used for making predictions.

2. Creating the Prediction Function:

Define a function that takes user input, processes it through the model, and returns the prediction. Gradio will use this function to generate predictions via a web interface.

3. Building the Gradio Interface:

Gradio allows you to define the input and output types simply, facilitating the design of the user interface for predictions.

4. Launching the Gradio App

Gradio provides a web interface that can run locally or be shared via a public link, enabling real-time interaction with the model.

## 1.4 API Integration

Gradio streamlines the process of converting a machine learning model into an interactive web application, which can also be exposed as an API for easy access. The following methodology outlines the steps involved:

1. Define the API Endpoint:

When launching a Gradio interface, it automatically serves an HTTP API. This API endpoint allows to send HTTP requests containing input data and receive model predictions as responses.

2. Input Formats:

Inputs to the API are typically sent as JSON objects. The structure of these JSON objects will depend on the input fields you define, such as numbers, text, or other data types.

3. Response Formats:

The model's predictions are returned as JSON objects. These responses can include predicted values or other relevant data points based on the model's output.

## 1.5 Security Considerations

No any security considerations were taken for the deployment.

## 1.6 Monitoring and Logging

The accuracy of the prediction is carried out using the Mean Squared Error which is integrated in the Gradio interface. No any logging mechanisms are integrated in the model.