National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Manoj Kanthraj
Student ID: 21218315

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Manoj Kanthraj
Student ID:	21218315
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	18/09/2023
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	3

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	18th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Manoj Kanthraj
21218315

1 Introduction

The effortless integration of machine learning algorithms and cloud computing has become essential for scalable and effective model deployment in today's data-driven environment. Combining the capabilities of platforms like Amazon Web Services (AWS) for deployment with those of Google Colab for algorithm development and improvement offers a powerful combination. This study provides a thorough methodology in which machine learning algorithms are initially developed and improved in the user-friendly Google Colab environment. The XGBoost algorithm's resilience is then utilized for AWS deployment via a Flask web application. This integration highlights the flexibility of cloud resources while also highlighting the significance of a seamless workflow from experimentation to actual implementation. The concomitant use of Postman to test APIs further demonstrates the end-to-end nature of the approach.

2 Prerequisites

The requirements for deploying the finalized model on AWS with Flask and Postman include proficiency in Google Colab, a firm understanding of fundamental machine learning concepts, familiarity with the ML models library, possession of an AWS account, proficiency in the Flask web framework, adept Python programming skills, basic web development understanding, and experience with tools like Postman for API testing. Making a good dataset for model training also requires expertise in data preparation, including feature engineering and data pretreatment. The required toolbox for a smooth implementation process is made up of these prerequisites.

3 Project Setup

3.1 Google Colab Development

Create a Google Colab notebook for model development

Import necessary libraries (NumPy, Pandas, Scikit-learn, XGBoost) and load your dataset.

Preprocess the data, perform feature engineering, and train all the machine learning models.

Tune hyperparameters and evaluate model performance.

3.2 AWS Setup

Create an AWS account if you don't have one.

Set up an EC2 instance: Choose an appropriate instance type, configure security groups, and generate a key pair for SSH access.

Connect to the EC2 instance using SSH and install necessary dependencies (Python, libraries, Flask).

Deploy your Flask app on the EC2 instance, ensuring it's accessible via the instance's public IP.

3.3 Flask Web Application

Install Flask using pip install Flask.

Create a new directory for your Flask project.

Design the Flask app structure: define routes, create API endpoints, and handle HTTP requests.

Integrate the trained finalized model into the Flask app.

Test the app locally using Flask run and verify the model's functionality.

3.4 Postman

Install Postman or use the web version.

Create a new request in Postman to test your deployed Flask API.

Send HTTP requests (GET or POST) with sample data and verify the responses.

3.5 Testing

Conduct thorough testing of the deployed Flask app, including finalized model functionality, API responses, and overall performance. Once satisfied, model is ready for real-world deployment and usage.

4 Project Implementation

The project is split into three notebooks in collab

"Agriculture_Sound_Classifier_Urban.ipynb"

"Agriculture_Sound_Classifier_Human_Voice_Detection.ipynb",

"Agriculture_Sound_Classifier_Full_AGri_Data.ipynb"

4.1 "Agriculture_Sound_Classifier_Urban.ipynb"

In this notebook, We concentrate on categorizing urban sound. We investigate and prepare an urban sound dataset, then test classification machine learning techniques. Our enhanced urban sound classifier, which offers insights into the distinctive soundscape of urban surroundings, is the result of model validation and fine-tuning.

Model	Features	Mean(STF_ACC)	Std_dev (STF_ACC)	VAL_ACC	Exec. Time (s)
RANDOM FOREST CLASSIFIER	MFCCS_CHROMA_MEL_CONTRAST_TONNETZ	90%	1%	90%	22.83
XGB CLASSIFIER	FFT_PSD_AR_MFCCS_CHROMA_MEL_CONTRAST_TONNETZ	87%	2%	88%	326.69

Figure 1: Result

4.2 "Agriculture_Sound_Classifier_Human_Voice_Detection.ipynb"

In this notebook, Now, human voice recognition is the focus of our attention. We examine a dataset with samples of human speech, preprocess the data, and apply algorithms. A powerful human speech detection is produced after careful model evaluation and optimization, demonstrating the promise of machine learning for identifying human vocalizations.

4.3 "Agriculture_Sound_Classifier_Full_AGri_Data.ipynb"

In this Notebook, With agricultural sound classification, the journey continues. We set up a sizable dataset of agricultural sounds, use data preprocessing and augmentation, and take advantage of XGBoost's modeling capabilities. We develop an agricultural sound classifier through careful training, evaluation, and improvement that is ready to decode the complex symphony of rural settings.

5 Evaluation and Results

The findings of XGBoost were the best when we looked at Validation accuracy and the mean of 5-Folds accuracy, but Decision Trees also produced results that were fairly comparable. Therefore, when execution time is considered, Decision Trees produced the outcome in just 0.04 seconds. Because of this, the various agricultural activities have extremely distinctive noises, and Decision Trees can be used to distinguish between them in real-time. If the sound signals are listened to, this can be heard and seen from above.

we can see that Random Forest achieves 90 percent accuracy in just 22.83 seconds, which is significantly faster than XGBoost's time of 326.69 seconds. XGBoost can identify both human voices and garbage in 0.81 seconds, but Random Forest takes 1.27 seconds to do it with just 85 percent accuracy. XGBoost can identify agricultural tasks with 97 percent accuracy in about 0.04 seconds, while Decision Trees can only get to 95 percent accuracy

Figure 1