Technical User Guide

Section 1 - York University, ML1030 - Capstone Project

1.1 Project

Kasi Insight Lending Environment and Lender Evaluation Tool

1.2 Description

This document provides step by step instructions on how to setup local development environment, handle dataset, and how to deploy the packaged application in AWS EC2 Instance.

1.3 Revision History

Date	Comment	Author
9/11/2019	Initial Draft	Vadim Spirkov
		Murlidhar Loka

Contents

Section 2 - Overview

2.1 Purpose

This document outlines the steps to complete so that the local development environment, generate models, clean dataset, creation of web application.

2.2 Scope

The details in this document are limited towards development and deployment functionality.

2.3 Project Artifacts

Project artifacts are available at Github . Download or clone the project repository before proceeding further.

Section 3 - Dataset Cleanup and Model Training Scripts

3.1 Prerequisites

Data cleaning/ imputation and model training scripts are developed employing Python and Python scientific libraries. Thus prior to starting working with the scripts Python 3.7 or newer has to be installed. To install Python follow this guide. Also make sure the *setuptools* package is installed. To install the package run *pip install -U setuptools*

After Python environment is installed following the steps listed below:

- Using command prompt tool navigate to the ..\src folder of the project. For Windows use cmd in admin mode.
- Run *pip install ./scripts*
- Verify that the kasi module is installed executing *pip list*. Make sure you see the package name in the list generated by the command

3.1 Data Cleaning and Imputation Process

The data processing script normalizes raw survey data. The output of the script is a comma-separated *.csv file that contains clean, imputed and categorised data.

The input of the scrip are raw KASI Insight survey data in *.csv format. The script can process many raw data files at once. The input files must meet the following requirements:

- They have to be comma-separated *.csv files with a header.
- The columns must have the same order as KASI survey does.

- The files has to be saved in UTF-8 format.

To execute the script run the following command in a command prompt processor of your choice:
kasi_process --input_dir [path_to_raw_data] --output_file [path_with_filename_to_store_clean_data]

To get more info about the script run *kasi_process -h*

Example: *kasi_process --input_dir C:/data/raw --output_file C:/data/processed/clean_data.csv*

3.2 Model Training Process

The model training script is very versatile. It can train both evaluator and simulator models employing a few Machine-Learning algorithms. The detailed information about the script could be found here.

This guide focuses on how to train production simulator and evaluator models. The input for the training script is the clean *.csv file generated at the previous step.

To train a **Lending Environment Simulator** model run the following command:

kasi_train --algorithm rf --model simulator --input_file [clean_data_dile] --output_dir [output_directory_name]

Example: *kasi_train --algorithm rf --model simulator --input_file ../clean/clean_data.csv --output_dir ./training*

To train **Lender Evaluator** model run the following command:

kasi_train --algorithm rf --model evaluator--input_file [clean_data_dile] --output_dir [output_directory_name]

Example: *kasi_train --algorithm rf --model evaluator --input_file ../clean/clean_data.csv --output_dir ./training*

The script generates a number of artifacts, namely:

- model image as a file with the extension *.model.
- model stats as a text file (extension *.txt)
- confusion matrix image (*.png file)
- learning curves image (*.png file)

Among other files the commands listed above will generate the following model files:

- Lender Environment Simulator model - **simulator_rf_tuned_large.model**
- Lender Evaluator model - **evaluator_rf_tuned_large.model**

Theses model files are to be deployed to cloud to do classification.

Section 4 - Backend Service API - Http Restful Service

The persisted models are utilized in the back end Flask based Web Application. The Restful API endpoints provides the ability to consume json input and provide the response as json - by utilizing the appropriate model.
The http response codes indicates if the input data is incorrect or if everything is as expected. Familiarity with Flask Microframework will help in maintaining the code base and/or future enhancements.

4.1 Build in Local Environment

4.1.1 Prerequisite

These steps depend on docker installed and configured.
If using windows 10 - install Docker Desktop Hub - this will provide "Kitematic" tool.
Code downloaded from GitHub and available in the local machine.

4.1.2 Build backend in local development windows machine

Launch docker VM by starting Kitematic docker tool.
Change directory to src/backend
Run the following commands

```
docker ps  ------- lists current docker images running
docker stop $(docker ps -a -q)  ----- stops all docker instances
docker rm $(docker ps -a -q)  ------ removes all the docker instances
docker build -t kasibackend:latest .   -----builds docker image with tag
docker run -p 5000:5000 -e PYTHONUNBUFFERED=0 kasibackend:latest --runs image
```

Note the ip address by running --- docker-machine ls

4.1.3 Test Restful API using Swagger and/or Postman tool

Access with browser: http://<ipAddress>:5000
This will launch Swagger UI for testing the api endpoints.
Follow the Swagger wizard to test the api endpoints
Postman tool can also be used to test the request to API

4.1.4 Push docker image to Docker repository

Private free repository can allow maximum of 1 image.
If more than one image is to be stored a paid account will need to be setup.
Log in on to docker hub using following command line from Kitematic:

```
docker login --username=yourhubusername --email=youremail@company.com
```

List docker images:

```
docker images
```

Tag docker image:

```
docker tag <ImageId>  <yourhubusername>/kasibackend:latest
```

Finally push the docker image to docker hub:

```
docker push <yourhubusername>/<Repository>
```

4.1.5 Download docker image from Repository

To pull an existing image from docker hub:

```
docker pull <yourhubusername>/<Repository>
```

4.1.6 Deploy downloaded docker image in server

We can run the following command to run the docker image:

```
docker run -p 5000:5000 -e PYTHONUNBUFFERED=0 kasibackend:latest
```

If docker image is found in local - it will be used - if not it will try to fetch from docker hub.

4.1.7 Save and load docker image as tar file

If constant upload and download of docker images needs to be avoided by saving image as tar file

```
docker load --input kasibackend.tar
```

Section 5 – AWS EC2 and Docker Images

5.1 Create AWS EC2 Instance

AWS EC2 instance can be created by selecting the appropriate AMI.
For our work - we utilized the free tier and restricted ourselves to ensure we are within the free tier usage limit. If this is going to be used for commercial purpose - appropriate level of security and capacity.
EC2 instance can either be created by aws cli or by using the aws console.
The step by step instruction is outlined in the following link:
https://docs.aws.amazon.com/efs/latest/ug/gs-step-one-create-ec2-resources.html
https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html

5.2 Build and Deploy AWS EC2 instance

To maintain the free tier requirement, we treated the AWS EC2 instances as regular linux servers.
The code base was moved from local environment to AWS EC2 instance.
The same steps were repeated to stop, remove existing docker images (if any).
Build and tag the docker image, followed by docker run - in following sequence:

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker build -t kasibackend:latest .
docker images
docker ps
docker run -p 5000:5000 -e PYTHONUNBUFFERED=0 kasibackend:latest
docker-machine ip
```

The above steps can be executed as a single shell script file in AWS EC2 instance.

Section 6 - Web Application FrontEnd (Angular)

The front-end of the project is developed using JavaScript, Angular 8 framework, CSS3 and HTML 5. It is expected that an experienced UI developer will be dealing with the project maintenance and enhancement. The UI project and brief Angular CLI command description could be found here.

6.1 Create and publish Front End Docker Image

The process of local deployment using Dockerfile is simillar to the steps outlined for backend local deployment.
Run the following commands in sequence:
Launch Kitematic CLI and change directory to location where frontend code is located
Update the proxy.conf.json file to point to the host where the backend api runs

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker build -t frontend:latest .
docker run -p 4200:4200 -e PYTHONUNBUFFERED=0 kasifrontend:latest
```

The docker image can be pushed/uploaded into docker hub if same needs to be shared.
As an alternative it can be saved as a tar file for later use for deployment.

6.2 Deploy in AWS EC2 instance

The frontend web application is deployed in its own AWS EC2 instance. The deployment steps are the same as outlined for backend with custom values for frontend application.
Update the proxy.conf.json file to point to the backend aws ec2 instance ip/host name

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker build -t frontend:latest .
docker run -p 4200:4200 -e PYTHONUNBUFFERED=0 kasifrontend:latest
```

Section 7 – Commercial AWS ECS managed Cluster for docker images

All the above steps can be totally managed by AWS ECS Fargate.
The ECS cluster, load balancer, proxy and backend api can be setup by using AWS Fargate and its aws command line interface. AWS will manage the entire setup and configuration.
The following link outlines the steps:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-cli-tutorial-fargate.html

Section 8 – References

Following references were used for this document:

Project Git Link:
https://github.com/v2msLabs/ML1030-Capstone-Project

Docker Essentials:
https://ropenscilabs.github.io/r-docker-tutorial/04-Dockerhub.html

AWS Step by Step Guide and tutorials:
https://docs.aws.amazon.com/efs/latest/ug/gs-step-one-create-ec2-resources.html
https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-cli-tutorial-fargate.html