Department of Computer Science and Engineering, University of Nevada – Reno

Gigafactory Systems Machine Learning Project

Team 15: Adam Cassell, Braeden Richards, Ashlee Ladouceur

Project Part 4: Progress Demo

Instructors: Sergiu Dascalu, Devrin Lee

External advisor(s): Dr. Emily Hand (UNR), Gavin Hall (Tesla)

14 March 2019

Table of Contents

Use Cases and Requirements Implemented

Use Cases

ID	Use Case	Description
UC01	DataInspection	Data Inspection tab allows user to see key statistics of dataset used to train model.
UC02	PreprocessData	User can prepare a model for training by selecting data from a data source and performing particular preprocessing actions on that data.
UC03	ModelLibrary	User can see Model Library that is filled with details about models in the System.

Table 1: Use cases implemented.

Requirements

Functional Requirements

ID	Requirement Description
FR01	System can normalize data.
FR02	System can standardize data.
FR03	System can input data from csv or xlxs file.
FR04	System can output accurate statistics of the data.
FR05	Interface can communicate with backend via GET/POST methods.
FR06	System will train and test models.
FR07	System can build models based off trained and testing data.
FR08	System can plot joint distributions for different features for comparisons.
FR09	System can train model according to specified number of epochs.
FR10	System can export model information as .h5 file.
FR11	User Inferface can read .h5 files for model information.

Table 2: Functional requirements implemented.

Non-functional Requirements

ID	Requirement Description
NF01	System shall be implemented using Python3.
NF02	System shall be Debian compatible.
NF03	System will use the TensorFlow library.
NF04	System interface will have a short user learning curve.
NF05	System interface will have intuitive design.
NF06	System shall maintain a simple user interface.
NF07	System shall accept input via mouse.
NF08	System shall accept input via csv and xlsx file.
NF09	System shall accept input via JSON file.
NF10	System will be able to output to csv file.
NF11	System shall be macOS and Windows compatible.
NF12	System will minimize system resource usage.
NF13	System is cross-browser and cross-platform.

Table 3: Non-functional requirements implemented.

Use Cases and Requirements To-be Implemented

Use Cases

ID	Use Case	Description
UC01	LogIn	User or admin can login with their information requested and enter an interface according to their permissions.
UC02	LogOut	User or admin can logout and the system will save all information modified.
UC03	AddModel	Admin can update the trained model into the Systems model library.
UC04	DiscardModel	Admin can discard the trained model.
UC05	TrainModel	Admin can enter training options manually and then enter that model into training.
UC06	StopTrain	Admin can halt training.
UC07	TrainSummary	Admin will be shown summary of training session after training is complete.
UC08	ModelPredict	The Model Predict tab will allow user or admin to test different inputs and outputs of model to see real-time prediction of the scenario.
UC09	ModelSwitch	User or admin can switch the model they are viewing on the system through the Model Library or toggling the model from the drop-down options.
UC10	ViewArchitecture	The Model Details tab subtab Summary displays the model architecture information accurately to the user or admin.
UC11	ViewStats	The Model Details tab subtab Inspect Data plots a visual of the statistics to the user or admin.
UC12	GetGuide	The GigaML User Guide button will link user or admin to internal wiki with user guide documentation maintained by users of the system.
UC13	ViewSystemDetails	The View System Information button will print accurate information regarding the factory systems, including parent and dependencies to the user or admin.

Table 4: Use cases to-be implemented.

Requirements

Functional Requirements

ID	Requirement Description
FR01	System will allow admins to have their own interface and permissions after sign-in.
FR02	System will allow users to have their own interface and permissions after sign-in.
FR03	System will enforce automatic timeouts during training.
FR04	System will take time series data and make predictions of that data for Model Predict tab.

Table 5: Functional requirements to-be implemented.

Non-functional Requirements

ID	Requirement Description
NF01	System shall accept user input via keyboard
NF02	System backend will have extensive error handling.
NF03	System will be able to output to xlxs file

Table 6: Non-functional requirements to-be implemented.

Summary of Project Status

Team 15 has made significant developmental progress in order to deliver a working deliverable for their scheduled progress demo on March 14, 2019. The development has been split between the three members equally. Adam worked on cleaning up the frontend for ease of user interaction. In addition, Adam worked on setting up a working model library and data inspection tab on the frontend that uses model information derived from the Python backend. The model library can now properly load objects from the backend, and once a model from the library is loaded, the data inspection tab is accurate according to data of that model. Braeden developed Python functions for preprocessing data from either csv's or excel files. Braeden also worked with Adam to create the communication between the backend and frontend. Finally, Ash created the functions in the Python script that take preprocessed data and train it with the Keras library. The models trained and tested from those functions are then saved into a folder that is referenced by the frontend for the model library on the frontend.

Communication between the frontend and backend has finally been achieved by using GET and POST methods. This cleared a major roadblock for the GigaML application. GET/POST methods have been chosen for ease of development with Flask. Instead of using forms on the frontend to send data through these methods, GigaML is using Ajax with JQuery to communicate since it is easier to use JavaScript functions in conjunction with Ajax than with forms. In summary, the frontend has a working Model Library, Data Inspection tab, and the Train tab accurately preprocesses data. The frontend does this by successfully communicating to the backend. The backend has most functions ready to go as the frontend comes together to satisfy future use cases to-be implemented.

Contributions of Team Members

Team 15 worked together to create the Progress Demo with the understanding of group effort. Multiple five-hour long meeting was held weekly between the team members prior to creating the document, so all members understood each section goal and outline project development. Table 7 is a representation of the time worked on each section:

Team Member	Coding/Sections	Time Worked
Adam Cassell	UC01-03, FR11, NF04-07, NF13, Use Cases and Requirements Implemented, Use Cases and Requirements To-be Implemented	35 hours
Ashlee Ladouceur	FR06-11, NF01-NF03, NF11-13, Use Cases and Requirements Implemented, Use Cases and Requirements To-be Implemented, Summary of Project Status, Contributions of Team Members	30 hours
Braeden Richards	FR01-05, NF01-02, NF08-13, Use Cases and Requirements Implemented, Use Cases and Requirements To-be Implemented	35 hours

Table 7: Detailed breakdown of time spent by each member of Team 15 for the Progress Demo.