

## SmartState

*Project Title: Event-driven Machine Learning, Intelligent Assessor (EMELIA)*

---

# Product Delivery User Manual

---

The User Manual is perhaps the most important document that the team will produce from the client's perspective. The user manual is the document the client will consult most going forward. A good user manual focuses on a single goal: educating the client on how to install, operate, and maintain the product.

### ***Team Members:***

David Rodriguez  
Reed Hayashikawa  
Andrew Hurst  
Jianxuan Yao  
Jesse Rodriguez

### ***Capstone Mentor:***

Fabio Marcos De Abreu Santos

Northern Arizona University  
*School of Informatics, Computing, and Cyber Systems*

### ***Clients:***

Jon Lewis  
Aaron Childers

*General Dynamics Mission Systems*

Version: 2.0  
Date: 5.10.2020



<b>1. Introduction</b>	<b>3</b>
<b>2. Installation</b>	<b>3</b>
2.1 Clone EMELIA from GitHub	4
2.2 Install Anaconda	4
2.3 Environment Setup	5
2.4 Run the Software	5
<b>3. Usage</b>	<b>5</b>
3.1 Train	6
3.2 Classify	6
3.3 Train & Classify	7
3.4 Help Text	7
<b>4. Maintenance</b>	<b>8</b>
4.1 Updating Anaconda Environment and Dependencies	8
4.2 State Management	9
<b>5. Troubleshooting</b>	<b>9</b>
5.1 Module Not Found Error	9
5.2 Error While Importing TensorFlow: Illegal Instruction	10
5.3 Import Error	10
<b>5. Conclusion</b>	<b>11</b>

# 1. Introduction

We are pleased that you have chosen EMELIA for your business needs. This event driven machine learning, intelligent assessor (EMELIA) is a software that is designed to assist system engineers in classifying system failures. Some of the key features of EMELIA include:

- The processing of files containing existing system fault reports known as tickets.
- The ability to train the machine learning model using previously classified tickets.
- The training model classifies ticket data using categories that have been used in previously classified tickets.
- The ability to report accuracy and performance information to the user.
- Saving the system's performance for a particular data set.

By using existing tickets to train our model, EMELIA will be able to accurately classify future tickets. This software will enhance the current workflow of General Dynamics by optimizing the classification of tickets and allow the system engineers to work more effectively. This user manual is to help you better install, manage, apply and maintain this program. Our goal is to ensure that the software is able to help you conserve resources so that you are able to profit from it as much as possible.

# 2. Installation

This section details the steps necessary to install the software and how to run EMELIA in the correct environment.

In this tutorial, we will cover the following steps:

- Clone EMELIA to a desired directory
- Install Anaconda 3.7 version from the Anaconda website
- Set up the environment
- Run the software

## 2.1 Clone EMELIA from GitHub

EMELIA is hosted online through the GitHub website. The site was used to have an accessible form of the project available for the development teams. The project version listed on the site is considered the canonical version of the project and represents the published form of the project as it is to be delivered to the client. To obtain the project:

1. Git will need to be installed on a local machine and the developer/user will need to have a GitHub account
2. A developer/user will also require collaborator permissions
3. Fork repository to user account
4. Use terminal/command prompt to navigate to the desired directory
5. Clone the repository to the user machine by running the command 'git clone <https://github.com/<GitHub Username>/Capstone-Project.git>' in the terminal or command prompt (depending on operating system).

The project has now been successfully installed on a local machine. Project development or source code analysis can begin by opening the project in a preferred text editor.

## 2.2 Install Anaconda

Anaconda is a package management system used to manage and deploy projects. Installation of Anaconda (Conda) is required and will require different steps depending on the operating system used by the developer or user.

- For Windows, follow these steps: <https://docs.anaconda.com/anaconda/install/windows/>
- For Linux, follow these steps: <https://docs.anaconda.com/anaconda/install/linux/>
- For macOS, follow these steps:  
<https://docs.anaconda.com/anaconda/install/linux/conda/install/mac-os/>

Once successfully installed, an Anaconda environment can be activated and dependencies can be installed via the Anaconda prompt.

## 2.3 Environment Setup

To begin:

1. Open an Anaconda prompt
2. Create a virtual environment by executing:

```
conda create -n <Environment Name> python=3.6 tensorflow=1.13.1 keras=2.2.4  
pyodbc=4.0.26 pandas=0.24.2 ipython=7.4.0 scikit-learn=0.20.3 numpy=1.16.5  
tqdm=4.46.0
```

- a. Replace <Environment Name> with preferred name of the environment
3. Activate the environment by typing "conda activate <Environment Name>"

The conda environment has been successfully setup and EMELIA can now be executed and source code can be developed.

## 2.4 Run the Software

1. Navigate to the directory where EMELIA is located
2. Once inside the project folder, change the current directory to the 'emelia' folder.
3. Please refer to the README.md document or the Configuration section to see possible commands.

## 3. Usage

Due to the nature of EMELIA as a machine learning system, there is little configuration to be made after downloading the code and creating the Anaconda environment. There are two basic use cases for this system, Training and Prediction.

## 3.1 Train

With the first use of EMELIA, training of a model is required. It is up to the discretion of the system engineers from there on out. It is generally good practice to retrain your model when any new labeled data is introduced to the system or if new kinds of labels or classifications are introduced into the error report type set.

The training of a model can be executed with the following in-line command:

```
python output.py --Train <AlarmData.csv> <TicketData.csv>
```

Where `<AlarmData.csv>` is the corresponding Alarm data file which contains the features, hex alarm code values, from the tickets being used to train with. An example of this file name would be, “AlarmData\_003.csv”. `<TicketData.csv>` is the corresponding Ticket data file containing the labels of each observation or ticket. An example of this file name would be, “TicketData\_003.csv”.

The CSV files used must be in the same directory as the output.py.

## 3.2 Classify

When a trained model exists, any unlabeled data can be passed through for its categorization to be predicted by the Neural Network. Run the following command:

```
python output.py --Classify <TestAlarms.csv> <AlarmData.csv> <TicketData.csv>  
<Predictions.csv>
```

Where `<TestAlarms.csv>` is the file containing the tickets that will be predicted for and `<Predictions.csv>` is the file the predictions will be written to.

The CSV files used must be in the same directory as output.py.

### 3.3 Train & Classify

The action of training a model and predicting on a file can be completed in one execution with the following command:

```
python output.py --Both <TestAlarms.csv> <AlarmData.csv> <TicketData.csv>  
<Predictions.csv>
```

Where <TestAlarms.csv> is the file containing the tickets that will be predicted for and <Predictions.csv> is the file the predictions will be written to.

The CSV files used must be in the same directory as output.py.

### 3.4 Help Text

To see a list of all of the previous commands you can run -h or --Help. This command will print instructions on the screen regarding current system commands and a description of those commands.

**Note:**

**The order of the filenames passed to the system are required. If files are passed in the wrong order, the system will not behave as expected.**

## 4. Maintenance

This section will highlight specific activities that can be completed in order to ensure the program can function at its most optimal settings. This section will include activities such as version and dependency control, upkeep of deprecated functions, and state management.

### 4.1 Updating Anaconda Environment and Dependencies

As our program is compiled of many external development sources and API's, maintenance on this component will be most likely needed. Being that we use anaconda for distribution of dependencies such as Tensorflow, scikit-learn, and numPy; these applications will need to be updated as new software releases are made. One way to indicate that these programs may need to be updated is to evaluate the warnings at runtime. The user will often find these warnings in the command prompt during the program's runtime and compilation. In order to address this, the user can simply open the anaconda prompt from their local PC and in the selected environment, run the command: `conda update --all`.

Along with updating all packages and versions, functions may be out of date due to the constant updates. Thus, if the user experiences any warnings in the command prompt such as deprecated function calls, the program will prompt the developer to change the following functions.

#### **Note:**

**If a function is deprecated, documentation will need to be searched to update usage of the Tensor or Keras API. If no update can be found, other options regarding implementation will need to be determined.**



## 4.2 State Management

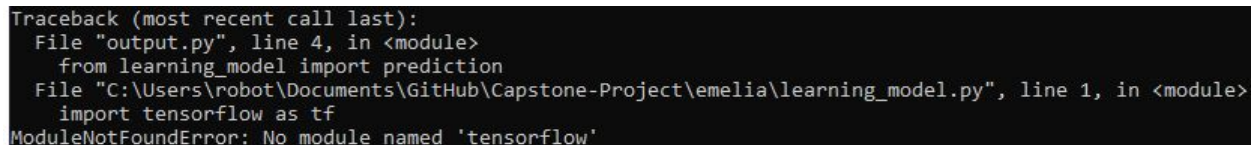
Emelia is composed of multiple neural networks that require constant training in order to get the optimal results. In the event that a new input value exists, the model needs to be retrained and the old model needs to be overwritten. The reason for this is because the model will try to predict the outcome for the unfamiliar input while using the old model and will cause the results to be incorrect. In order to prevent this, the program must undergo training periodically in order to ensure every input value is being accounted for.

## 5. Troubleshooting

This section details what errors that might occur when setting up this software environment. The most common errors have to do with the dependencies in the software environment, misuse of imports, or attempt to run the program without activating an Anaconda environment.

### 5.1 Module Not Found Error

Screenshot:



```
Traceback (most recent call last):
  File "output.py", line 4, in <module>
    from learning_model import prediction
  File "C:\Users\robot\Documents\GitHub\Capstone-Project\emelias\learning_model.py", line 1, in <module>
    import tensorflow as tf
ModuleNotFoundError: No module named 'tensorflow'
```

This error can be caused by a dependency (such as tensorflow) not installing correctly when the environment is set up. This error can occur with any of the dependencies within the environment. The solution to this issue is to double check the environment dependencies using “conda list” to make sure that the correct version of tensorflow is installed. Another useful step is to run the command “conda remove tensorflow” within the Anaconda prompt with the environment activated. After confirming that you would like to proceed, wait until it finishes uninstalling then run the command “conda install tensorflow=1.13.1” to reinstall the dependency with the

correct version. You may find it necessary to repeat these steps for other dependencies once one issue is resolved.

Another source for this error can be that a user of the system attempts to run the system without activating their Anaconda environment. This the easiest issue to resolve, and can be done by running the command: `conda activate <Environment Name>`

## 5.2 Error While Importing TensorFlow: Illegal Instruction

This error is caused by not having sufficient hardware to utilize tensorflow.

The solution to this issue is to transfer the software to a newer computer that meets the basic requirements of tensorflow and running the software again. Alternatively, setting up a virtual environment may also solve this issue but it is not recommended.

Here are the System requirements for Tensorflow:

- Python 3.5–3.7
- pip 19.0 or later (requires manylinux2010 support)
- Ubuntu 16.04 or later (64-bit)
- macOS 10.12.6 (Sierra) or later (64-bit) (*no GPU support*)
- Windows 7 or later (64-bit) (*Python 3 only*)
  - Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019
- Raspbian 9.0 or later
- GPU support requires a CUDA®-enabled card (*Ubuntu and Windows*)

For all inquiries related to hardware needed to run EMELIA, please refer to ‘Appendix A’ of the document titled ‘Final Report’.

## 5.3 Import Error

Screenshot:

```
ImportError: DLL load failed: The specified procedure could not be found.
```

This issue is caused by a separate version of python installed on the system and the version used by anaconda.

The solution to this issue is to use the command “pip uninstall <dependency>” and then “pip install <dependency>” for each dependency where this issue occurs. This will create a new instance of the package you are having trouble with and allow anaconda to use the packages when running the software.

## 5. Conclusion

Overall, the purpose of this document is to assist you in setting up and performing the main functionality of EMELIA. We hope that you have many happy years of increased productivity with our product. With best wishes from your EMELIA developers: David Rodriguez (dr475@nau.edu), Reed Hayashikawa (rmh325@nau.edu), Andrew Hurst (abh266@nau.edu), Jesse Rodriguez (jkr232@nau.edu), and Jianxuan Yao (jy394@nau.edu). As we all move on to our own professional careers in the future, we hope you continue to reach out to us if you have any remaining questions on how to optimally operate EMELIA. It has been a pleasure to work with you, thank you for the opportunity.