PREDICTING PILOT PERFORMANCE BASED ON TIME-SERIES PHYSIOLOGICAL AIRLINE PILOT DATA

GEORGE MASON UNIVERSITY

VOLGENAU SCHOOL OF ENGINEERING

SPRING 2019

BENITEZ AGÜERO, ROMINA

BESSIN, KOFFI PAULE

GURIJALA, SIDDHARTHA REDDY

MITTAL, RADHIKA

OTOO, JOHN DAVID

TABLE OF CONTENTS

Abstract

Automation implemented in aircrafts faces safety challenges as pilots' direct involvement is reduced, in this way, directly interfering with the reaction time needed to correct a potential issue and ultimately putting the lives of the crew and passengers in danger. Recent aircraft accidents have put in question the advantages that automation brings into a flight, especially in how it affects a pilot's level of engagement. Relegating a pilot to a monotone task would place the pilot in an undesirable cognitive state of distraction that will reduce the reaction time available in case of failure in the systems.
Data from 18 pilots in 9 crews exposed to 3 different controlled experiments (Channelized Attention (CA), Diverted Attention (DA), or Startle/Surprise (SS)), was used to perform Time series analysis by means of Long short-term memory (LSTM). The model predicts the likelihood that a pilot would be in a specific cognitive state in the next second by using the most recent 35 seconds readings corresponding to 8,960 physiological records. These predictions can then be displayed via an alerting system that will let the pilot know when in a state of diverted attention lasting for more than 5 seconds so that they can take the necessary actions to take control of the situation. This alerting system provides a solution to the risks brought by automation in aircrafts' cabins, thus effectively reducing the chance of occurrence of accidents and fatalities.

1. Introduction

1.1. Background and Rationale

Airplanes are one of the most complex transportation machines present on earth right now. It makes life much easier for a lot of people. Considering that, the person behind the wheels (pilot) has a lot of stress associated with their job. For that matter, a google search of “what is the most stressful job” ranked Airline pilot job as the 3rd hardest and the stressful situation on earth.

It has been proven that the pilot goes through a maximum of stress during the takeoff and landing which for a reason sounds valid because during the rest of the time the flight automated. With the automation available today a flight can be set into auto-pilot mode in less than 5 secs after the takeoff.

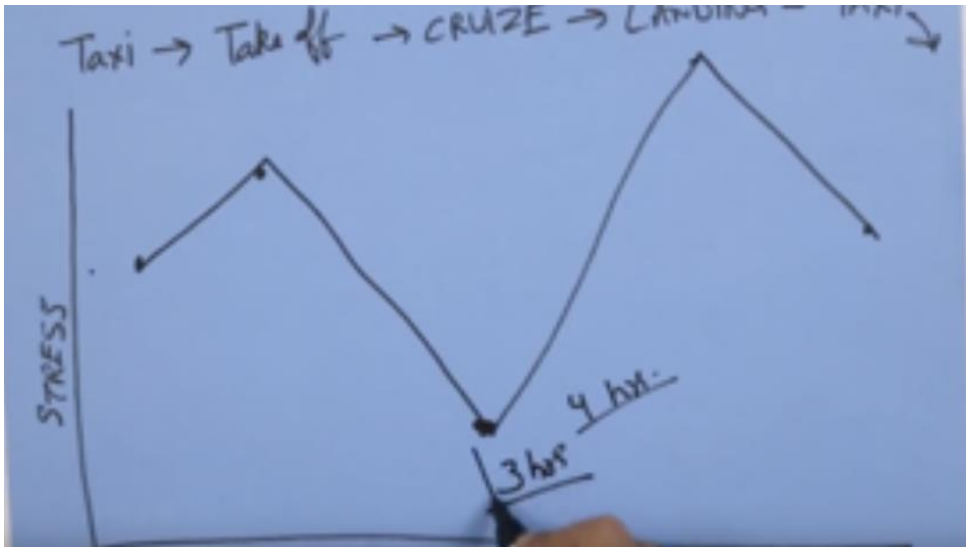


Figure 1. Pilot stress graph

In regard to automation, the two industry leaders have two entirely different approaches towards the pilot automation process. Boeing believes that automation is a good thing and should be implemented, but Humans are supposed to be kept in the loop whereas its rival Airbus thinks that Computers should be given priority and let it do things to make humans work simpler.[1] This can be understood better from the Flight crash incidents of Air France 477 and Turkish Airlines TK 1951.

Air France Flight 447 was in route from Rio de Janeiro to Paris on May 31, 2009, for an overnight trip, when it vanished. There was no warning before the flight crashed into the Atlantic Ocean in the early morning hours of June 1, 2009 -nearly four hours after take-

[1] Little, B. (2019, March 21). *Automation of Planes Began 9 Years After the Wright Bros Took Flight—But It Still Leads to Baffling Disasters.*Retrieved from HISTORY: https://www.history.com/news/plane-automation-autopilot-flight-302-610

off. The 2009 crash, which killed all 228 passengers and crew on board, is considered one of the worst -- and most mysterious -- aviation disasters in modern history. It has been proven that the co-pilots were not adequately trained and depended too heavily on the plane's automated system. There was a slight diversion of the pilot's state of mind and raised the right stick which made the plane to stall.[2]

Figure 2. Overview of the AF 447 Accident

Now coming into the second incident. Tk 1951 The aircraft, a Turkish Airlines Boeing 737-800, crashed into a field approximately 1.5 kilometers north of the Polderbaan runway, (18R), before crossing the A9 motorway inbound, at 09:26 UTC having flown from Istanbul, Turkey. The aircraft broke into three pieces on impact. The wreckage did not catch fire. The crash was caused primarily by the aircraft's automated reaction, which was triggered by a faulty radio altimeter. This caused the auto throttle to decrease the engine power to idle during approach. The crew noticed this too late to take appropriate action to increase the thrust and recover the aircraft before it stalled and crashed. This can be observed as no proper attention was given by pilots to address the issue. All these crashes explain us the importance of the presence of mind a pilot should have and the tracking that state of mind of a pilot and alerting them needs to be considered as an important factor in today's automated aviation. The project we are working on helps us to achieve that.

1.2 Research

The Kaggle competition related to our project stems from an ongoing study within the extensive ongoing research focused on predicting pilot error, especially within the Federal Aviation Administration (FAA) Next Generation Air Transportation System (NextGen)

[2] (Bureau d'Enquêtes et d'Analyses(French Civil Aviation Safety Investigation Authority). (2012). *Final Report On the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro - Paris.*France: BEA.

program, a multibillion-dollar modernization of the National Airspace System (NAS). Targeted for full modernization to be in place by 2025, this system will impose revolutionary procedures and technology on the air transport triad (i.e. Flight deck pilots, Air traffic controllers and automation). These changes must be assessed, not only for technical assurance and reliability, but for feasibility in operational context. Not only is there general concern as to whether the added procedures and responsibility may lead to pilot workload overload, but, additionally, it is a concern that the layers of automation designed to mitigate this overload may degrade vs. enhance the situation awareness on the flight deck.[3]

As part of this effort, research has been made in Predicting Pilot Performance in Off-Nominal Conditions that has shown how a computational model of noticing might predict data from a realistic flight simulation, based on the observed probability of noticing off normal events that are either surprising or unexpected. The model predicted the magnitude of three effects of considerable importance to aviation safety: the reduced detection of unexpected events, the "attentional tunneling" effects of the HITS head down on the instrument panel, and the general advantage of detecting rare events in the forward view, rather than head down.[4]

Finally, the research which we were talking earlier seeks to validate a safety performance index as a tool based on quantification of meaningful aviation safety system properties, with the potential to grasp the intangible domain of aviation safety. The tool itself was developed in the form of Aerospace Performance Factor and is already available for the aviation industry. However, the tool impact was considered unsuccessful, as its potential was not fully recognized by the industry.

The sponsor of the project is focused on this goal, and the goal of this project(Pilots state of mind prediction) is to hypothesize and validate the patterns the data supports and to help categorize what else the data might be saying reliably, during this project our team performed analysis on the potential and outlines new features, utilizing time-series analysis, which can improve both the recognition of the index by the industry as well as the motivations to further research and develop methodologies to evaluate overall aviation safety performance using its quantified system properties. The paper summarized the options for applying robust time-series analysis for the purpose of safety performance index prediction. Our ambition is to shift the existing ideas and approaches proposed by other Kaggle competitors, in order to explore new ways for achieving future predictive risk management, by exploiting both mathematical capabilities and recent safety engineering practice and principles. [5]

---

[3] Wickens, C.D., Sebok, A., Gore, B.F., & Hooey, B.L. (2012). Predicting Pilot Error in Nextgen: Pilot Performance Modeling and Validation Efforts.

[4] Wickens, C., Hooey, B. & Gore, B., Sebok, A., Koenecke, C. & Salud, E. (2009). Predicting Pilot Performance in Off-Nominal Conditions: A Meta-Analysis and Model Validation. Human Factors and Ergonomics Society Annual Meeting Proceedings. 53. 86-90. 10.1518/107118109X12524441079102.

[5] Lalis, A. (2017). Time-series analysis and modelling to predict aviation safety performance index. Transport Problems. 12. 51-58. 10.20858/tp.2017.12.3.5.

1.3 Motivation

Despite the fact that airplane crashes attract great attention from the media, air travel is widely accepted as safer than highway travel and with good reason. On one of their most recent reports, the National Highway Traffic Safety Administration states that in 2016, there were 34,439 fatal crashes from the total 7,277,000 reported motor vehicle crashes. Less than 0.5%[6]. In comparison, The National Transportation safety board reports for Civil Aviation 0 fatal accidents for air carriers out of the 1,335 accidents recorded for the year 2016[7]. Additionally, the recent Lion Air and Ethiopian Airlines crashes have called into question the process by which autonomous sub-systems in commercial aircrafts can be determined as safe and trusted within the operational context of the aircraft.

Pilot awareness measures reveal that less thinking related to the flight is taking place when there is more automation in the cockpit. A 2013 study found that pilots using an automated cockpit that seemed to be having no issues, resulted in pilots losing focus and thinking about a topic unrelated to managing the plane[8]. When there is a fault in the system, the reaction time of a pilot can decide the fate of the aircraft and its passengers, but distracted pilots can take some time to realize an alarm has gone off [9] [10]. Monitoring the state of awareness of pilots in real time and producing a prediction of whether they seem to be getting distracted can grant valuable seconds that can be used to properly react in the event of failure of a system. An alerting system of this kind going off in both the cockpit and the Air Traffic Control tower could help keep the pilots focused and thus prevent aviation fatalities.

1.4 Project Objectives

To provide data and evidence supporting the feasibility and safety value of an implementing and installing an alerting system which provides pilot(s) with information about in-flight cognitive status allowing them to correct mistakes on time and ultimately prevent aircraft accident and fatalities.

---

[6] National Center for Statistics and Analysis. (2018). Police-reported motor vehicle traffic crashes in 2016. (Research Note Report No. DOT HS 812 501). Washington, DC: National Highway Traffic Safety Administration.https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812501

[7] National Transportation Safety Board (2018). 2016 NTSB US Civil Aviation Accident Statistics. https://www.ntsb.gov/investigations/data/Pages/AviationDataStats2016.aspx

[8] Casner, S. M., & Schooler, J. W. (2014). Thoughts in Flight: Automation Use and Pilots' Task-Related and Task-Unrelated Thought. Human Factors, 56(3), 433–442. https://doi.org/10.1177/0018720813501550

[9] Kitroeff, N., Gelles, D., Glanz, J, & Beech, H. (2019). Ethiopian Crash Report Indicates Pilots Followed Boeing's Emergency Procedures. The New York Times. https://www.nytimes.com/2019/04/04/business/boeing-737-ethiopian-airlines.html

[10] 1999, Sep 15. Pilots Slow to React in Buenos Aires Crash. The New York Times. https://www.nytimes.com/1999/09/15/world/pilots-slow-to-react-in-buenos-aires-crash.html

1.5. Problem Space

Leverage EEG data to predict when a pilot or co-pilot may go into one of the three following cognitive states and to design an alerting system

- Channelized Attention (CA)
- Diverted Attention (DA)
- Startle/Surprise (SS)

1.6. Primary User Story

As a user, I want to have a detailed analysis of the sequence of events leading up to airline accidents so that several opportunities for a pilot to focus and intervene are identified through an alerting system. The absence of intervention by pilots is considered to be the result of a "loss of "airplane state awareness." Ineffective attention management on the part of pilots who may be distracted, sleepy or in other dangerous cognitive state may lead to accidents.

1.7. Solution Space

Our system delivers value when it accurately predicts a pilot or co-pilot entering an state of "loss of awareness" and alerts them so that they can correct back to a focused state (channelized attention). Users derive value from these predictions as they will have more time to manually pilot the aircraft and assess the situations they are found in while they are in an adequate cognitive status. In this way, going through protocol emergency checklists and gaining control of the aircraft is more likely to happen in a time-effective manner. We expect that ultimately, our system, will prevent accidents and fatalities in commercial flights.

1.8. Product Vision

Table 1 shows the potential usability the finalized product may have.

1.9. Definition of Terms

- **Electroencephalogram:** signals of electrical activity of the brain.[11]
- **Electrocardiogram:** signals of electrical activity of the heart.[12]
- **Galvanic Skin Response:** electrodermal activity.[13]
- **Principal component analysis**: a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.[14]
- **Linear discriminant analysis:** a type of linear combination, a mathematical process using various data items and applying functions to that set to separately analyze multiple classes of objects or items.[15]

[11] Retrieved from https://www.webmd.com/epilepsy/guide/electroencephalogram-eeg

[12] Retrieved from https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983

[13] Retrieved from https://en.wikipedia.org/wiki/Electrodermal_activity

[14] Retrieved from https://en.wikipedia.org/wiki/Principal_component_analysis

[15] Retrieved from https://en.wikipedia.org/wiki/Linear_discriminant_analysis

	User Sample Scenario 1	User Sample Scenario 2
For	National center for air transportation	Commercial Airlines
Who	are focused to increase the security of aircrafts and make air a safer channel of transportation.	are focused on increasing profit by marketing higher safety standards than government mandated regulations.
The	prediction system	prediction system
Is a	Accident Prevention System	Alerting system
That	is used to predict changes in the cognitive status of a pilot and produce an alert.	will predict when a pilot or co-pilot (user) will go into an unfavorable cognitive state and emit an alert
Unlike	products not tested and approved by FAA for commercial use yet	aircraft sensors that may malfunction
Our product	would Predict any change in pilots' states of mind	will rely on the awareness state of the pilot.
Caveats	the success of the system assumes that upon being alerted, the pilots will use their new cognitive state to address any issues related to the aircraft and adequately pilot manually. The system might not produce an alerting signal when the reading is "channelized attention" but this does not guarantee the attention of the pilot is directed to the aircraft. Pilot may be very focused in a different activity.	the success of the system assumes that upon being alerted, the pilots will use their new cognitive state to address any issues related to the aircraft and adequately pilot manually. The system might not produce an alerting signal when the reading is "channelized attention" but this does not guarantee the attention of the pilot is directed to the aircraft. Pilot may be very focused in a different activity.

Table 1. Side by side sample users of our product

2. Data Acquisition

2.1. Overview

The datasets used in this project were originated by Booz Allen Hamilton and made available through Kaggle.

- Test Data: 4 GB, 4867421 entrees and 28 variables
- Train Data: 1.2 GB, 17965143 entrees and 28 variables

2.2. Field Descriptions

Field Name	Data Type	Description	Example
crew	Integer	A unique id for a pair of pilots. There are 9 crews in the data.	1
experiment	String	One of CA, DA, SS or LOFT. The first 3 comprise the training set. The latter the test set.	CA
time	Decimal	Seconds into the experiment	10
seat	Integer	Identifes if pilot subjucted to experiment is in the left (0) or right (1) seat	0
eeg_fp1	Decimal	Electroencephalogram recording. Frontopolar electrode.	0.643204
eeg_f7	Decimal	Electroencephalogram recording. Frontal electrode.	-0.912441
eeg_f8	Decimal	Electroencephalogram recording. Frontal electrode.	5.46683
eeg_t4	Decimal	Electroencephalogram recording. Temporal electrode.	11.4204
eeg_t6	Decimal	Electroencephalogram recording. Temporal electrode.	4.15584
eeg_t5	Decimal	Electroencephalogram recording. Temporal electrode.	4.4678
eeg_t3	Decimal	Electroencephalogram recording. Temporal electrode.	0.859876
eeg_fp2	Decimal	Electroencephalogram recording. Frontopolar electrode.	1.20827
eeg_o1	Decimal	Electroencephalogram recording. Occipital electrode.	6.40349
eeg_p3	Decimal	Electroencephalogram recording. Parietal electrode.	4.21895
eeg_pz	Decimal	Electroencephalogram recording. Parietal reference electrode.	12.1544
eeg_f3	Decimal	Electroencephalogram recording. Frontal electrode.	2.1919
eeg_fz	Decimal	Electroencephalogram recording. Frontal reference electrode.	4.44127
eeg_f4	Decimal	Electroencephalogram recording. Frontal electrode.	4.70046
eeg_c4	Decimal	Electroencephalogram recording. Central electrode.	8.80657
eeg_p4	Decimal	Electroencephalogram recording. Parietal electrode.	11.8771
eeg_poz	Decimal	Electroencephalogram recording. Pariental-occipital junction reference electrode.	8.65578
eeg_c3	Decimal	Electroencephalogram recording. Central electrode.	7.57234
eeg_cz	Decimal	Electroencephalogram recording. Central reference electrode.	8.3905
eeg_o2	Decimal	Electroencephalogram recording. Occipital electrode.	7.65356
ecg	Decimal	3-point electrocardiogram signal. Units= microvolts	-18320.699
r	Decimal	Respiration, a measure of the rise and fall of the chest. Units= microvolts	666.06799
gsr	Decimal	Galvanic Skin Response, a measure of electrodermal activity. Units= microvolts	834.43103
event	String	The state of the pilot at the given time: one of A = baseline, B = SS, C = CA, D = DA	C

Table 2. Data dictionary

2.3. Data Context

The dataset provides real physiological data from eighteen pilots who were subjected to various distracting events. The training set summarizes collected data from the pilots experiencing different experiments in a controlled non-flight environment. These experiments have for goal to induce three specific types of distractions and generate one of the cognitive states that will be explained. The test set, just like the training set summarizes data from the same experiments however, these experiments are recorded in a full flight simulator. The three different cognitive states aforementioned, are defined as following:

- Channelized Attention (CA) is, roughly speaking, the state of being focused on one task to the exclusion of all others. This is induced in benchmarking by having the subjects play an engaging puzzle-based video game.
- Diverted Attention (DA) is the state of having one's attention diverted by actions or thought processes associated with a decision. This is induced by having the subjects perform a display monitoring task.

Periodically, a math problem showed up which had to be solved before returning to the monitoring task.

- Startle/Surprise (SS) is induced by having the subjects watch movie clips with jump scares.[16]

2.3.1. Data Conditioning-Preprocessing

To condition our datasets, different techniques were considered to facilitate the application of our machine learning models.

- Data Cleaning

To clean our data, we checked for mismatched or misclassified data, outliers and patterns of missing data. Our evaluation showed that we are dealing with fairly cleaned datasets. We accounted no missing values, or misclassified/mismatched data. The cleaning process for these datasets focused on removing outliers from the electroencephalogram (EEG) readings. EEG observations that were 4 standard deviations away or more from the mean were deleted, thus fetching us cleaned data.

- Dimensionality Reduction

To find which were the best predictors for our analysis, a dimensionality reduction technique was applied to our data. We explored Principal Component Analysis (PCA) to keep the variables with the highest variance out of the original 28 features in the original training data seen in figure 3. However, because we are performing a time series analysis on our data, we decided to discard PCA as it was ignoring the true meaning of the time stamp present in our data.

crew	exp	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_o1	eeg_p3	eeg_pz	eeg_f3	eeg_fz	eeg_f4	eeg_c4	eeg_p4	eeg_poz	eeg_c3	eeg_cz	eeg_o2	ecg	r	gsr	event
1	DA	106.3086	0	-3.98208	0.120569	-10.7179	-12.2361	-24.8031	-1.0158	-8.96937	-17.6215	1.4217	-0.74484	1.68985	-1.86746	-11.07	-15.2462	-4.93541	-7.76079	-0.93961	0.872699	-4.08357	-5.99492	0	656.046	702.426	D
1	DA	106.3125	0	-10.5966	-7.27848	-26.7922	-29.5004	-20.7133	-2.44403	-7.11496	-18.9916	4.0876	3.30375	4.4284	-4.99332	-10.5822	-19.228	-9.45285	-3.56534	2.35787	0.227763	-4.37896	-3.89871	0	656.046	702.426	D
1	DA	106.3164	0	-15.5952	-6.13584	-26.0077	-2.16881	-10.4986	1.11091	-10.6323	-27.8484	2.73442	-0.97172	3.12646	0	-9.01556	-7.81245	1.58731	-2.39646	1.06411	-0.25739	-4.40924	-3.94037	0	656.046	702.426	D
1	DA	106.3203	0	-9.28737	1.47248	-13.1929	-9.55804	-18.3331	-4.8216	-7.15265	-18.1887	1.0861	-3.35275	-0.36997	7.70405	-6.38421	-3.90611	-3.38464	-3.97815	-0.18208	-2.91605	-6.48904	-4.4709	0	656.046	702.426	D
1	DA	106.3242	0	-6.28109	-3.1302	-13.4015	-11.1376	-12.4419	-5.42116	-16.0233	-9.22404	1.51536	-5.34321	-1.29021	-6.48652	-7.86349	-6.40476	-5.82083	1.79472	-0.9283	-5.9024	-10.0009	-1.77807	0	656.046	702.426	D
1	DA	106.3281	0	6.97473	5.55771	-7.94278	-1.20956	-17.666	2.45509	6.62475	5.29681	1.73893	-2.15922	-0.47536	-2.80985	-5.05401	-6.08264	-2.7577	-1.00848	-1.17478	3.28883	-6.56104	-1.56003	0	656.046	702.426	D
1	DA	106.332	0	1.96606	1.35491	-15.4411	-10.3826	-12.7311	2.00397	3.34673	0.682138	-1.35487	-6.28416	-5.89351	-4.72802	-11.7224	-7.35525	-10.6744	-9.61922	-4.7081	-4.50424	-11.9765	-5.72311	0	656.176	702.503	D

Figure 3. Snippet of training dataset showing 28 features

- Data Balancing

Our dataset is imbalanced since there are maximum number of records in the CA state of event, so to deal with this we initially used the SMOTE function to deal with the imbalance. This package allowed us to resample our data and therefore balance it.

- Data Aggregation

As part of our data preprocessing actions, we aggregated the data into 1 second windows. this step would significantly reduce the size of our data. In the original dataset, there was about 4,867,421 records and 28 variables. this translated to 257 reading every second or one reading about every 0.0038

[16] Retrieved from https://www.kaggle.com/c/reducing-commercial-aviation-fatalities/data

second. As part of the team's data preprocessing, we decided to aggregate the data into 1 second windows. Tis step reduced the count of records to about 505,660. this was an 89% reduction in the data size. This step helps us utilize tools such as WEKA to further explore the data.
The drawback of this step was the loss of data. When aggregating data (i.e average and median) the likelihood of losing the value of each data point is elevated but as outliers may skew the aggregations. To mitigate this the team addressed the likely issue of outliers before aggregating the dataset.

crew	Exp	seat	Count	in seconds	in minutes
1	CA	0	92131	359.8867	5.998112
		1	92168	360.0313	6.000521
	DA	0	92077	359.6758	5.994596
		1	92130	359.8828	5.998047
	SS	0	39563	154.543	2.575716
		1	39583	154.6211	2.577018
2	CA	0	92133	359.8945	5.998242
		1	92099	359.7617	5.996029
	DA	0	92194	360.1328	6.002214
		1	92099	359.7617	5.996029
	SS	0	92131	359.8867	5.998112
		1	92212	360.2031	6.003385
3	CA	0	92137	359.9102	5.998503
		1	92095	359.7461	5.995768
	DA	0	92133	359.8945	5.998242
		1	92141	359.9258	5.998763
	SS	0	92132	359.8906	5.998177
		1	92157	359.9883	5.999805
4	CA	0	92200	360.1563	6.002604
		1	92137	359.9102	5.998503
	DA	0	92164	360.0156	6.00026
		1	92137	359.9102	5.998503
	SS	0	92159	359.9961	5.999935
		1	92084	359.7031	5.995052
5	CA	0	92137	359.9102	5.998503
		1	92115	359.8242	5.99707
	DA	0	92162	360.0078	6.00013
		1	92130	359.8828	5.998047
	SS	0	92134	359.8984	5.998307
		1	92137	359.9102	5.998503

crew	Exp	seat	Count	in seconds	in minutes
6	CA	0	92157	359.9883	5.999805
		1	92097	359.7539	5.995898
	DA	0	92253	360.3633	6.006055
		1	92136	359.9063	5.998438
	SS	0	92161	360.0039	6.000065
		1	92154	359.9766	5.999609
7	CA	0	92132	359.8906	5.998177
		1	92106	359.7891	5.996484
	DA	0	92135	359.9023	5.998372
		1	92100	359.7656	5.996094
	SS	0	92160	360	6
		1	92136	359.9063	5.998438
8	CA	0	92108	359.7969	5.996615
		1	92136	359.9063	5.998438
	DA	0	92108	359.7969	5.996615
		1	92087	359.7148	5.995247
	SS	0	90762	354.5391	5.908984
		1	90758	354.5234	5.908724
13	CA	0	92162	360.0078	6.00013
		1	92126	359.8672	5.997786
	DA	0	92078	359.6797	5.994661
		1	92129	359.8789	5.997982
	SS	0	92136	359.9063	5.998438
		1	92093	359.7383	5.995638

Table 3. Training dataset experiments

2.3.2. Data Quality Assessment

- Completeness: Data has no missing values NA's, none of the data features are in an unusable state. Data is comprehensive.
- Uniqueness: Data does not have a set of redundant values and given the data source is unique for this problem statement.
- Accuracy: Data correctly reflects the events occurring with respect to the pilot under observation.
- Atomicity: Data shows granular timely values of the EEG, ECG, GSR.

- Conformity: Data follows the standard set of data definition, desired formats with time stamps and character variables.
- Overall Quality: Good quality data.

2.3.3. Other Data Sources

Alternative sources of physiological data to train our model may be beneficial and a recommended approach for future work due to the limited amount of time available we had to develop this project. A potential source of EEG data is Center of the Institute for Neural Computation at the University of California San Diego which not only gives access to data collected by them, it also shares other potential sources of EEG readings. [17]

3. Analytics and Algorithms

3.1 Prior analysis

As an open Kaggle competition at the time this project was started, several teams from around the world have tried different classification machine learning methods looking to find the best fitted model that will produce a high accuracy rate when predicting a pilot's' state of mind during a full simulated flight. Several algorithms have been tried and decent results were achieved, many of them publicly documented on the Kaggle's competition site.

3.2 First Approach

Following the teams who participated in the Kaggle competition approach, our team decided to explore several machine learning techniques for our analysis as well. The initial goal was to explore few classification algorithms such as XGBoost, Dynamic Time Warping, Support Vector Machine (SVM), Cox Regression, Naive Bayes and Random Forest. Amongst these algorithms, three techniques have been chosen to be implemented and from the results obtained, we were able to compare different metrics. These three techniques implemented were XGBoost, Random Forest and SVM algorithms. Table 4 below shows the advantages of using these techniques on our huge dataset to predict the pilots' state of minds.

Models	**Advantage 1**	**Advantage 2**	**Advantage 3**
XGBoost	Handles High Dimensional Data	Fast Processing	Regularization
Random Forest	Handles High Dimensional Data	Decorrelates trees	Regularization
SVM	Handles High Dimensional Data	Performs well with non-linear boundary	Maximizes Margin between classes

Table 4. Comparison of models

Figure 4 and Figure 5 are snippets of the outputs obtained from running the model, the accuracy and the confusion matrix, respectively.

[17] Retrieved from https://sccn.ucsd.edu/~arno/fam2data/publicly_available_EEG_data.html

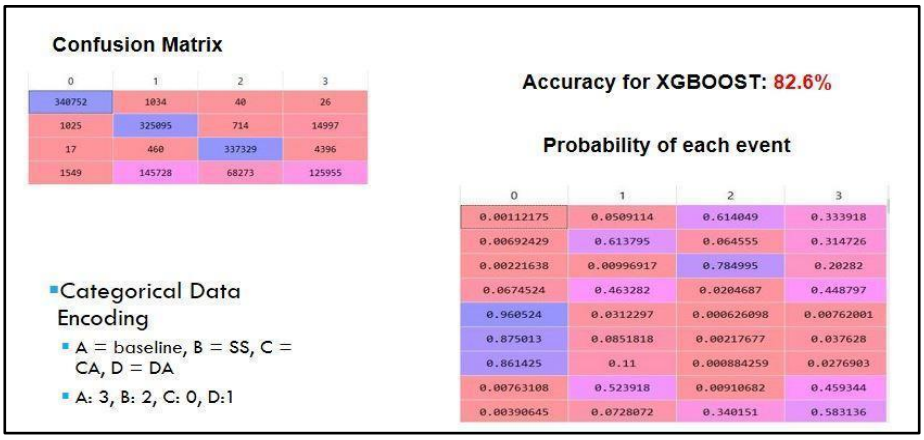


Figure 4. XGBoost Accuracy Results

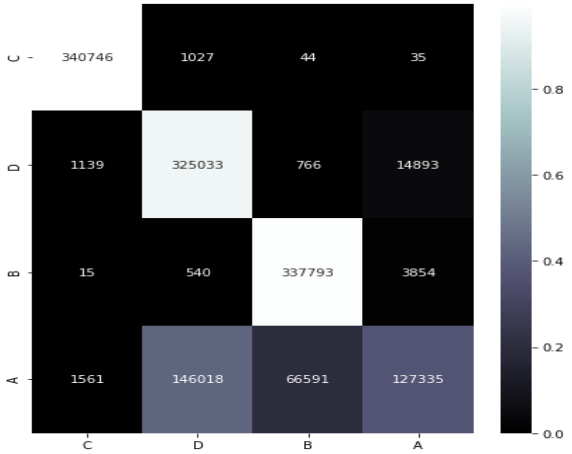


Figure 5. XGBoost Confusion Matrix

3.2.1. XGBoost

XGBoost which stands for eXtreme Gradient Boosting, is a powerful and advanced gradient boosting algorithm. The choice to use this algorithm for analyzing our data, is its fast processing ability and regularization implementation that helps reduce overfitting. Before building the model, the initial train (training) dataset provided by Kaggle was reduced. We decided to use a sample of 40% to perform our analysis. The model was then trained on that new training set using respectively 70% for training and 30% for validation. 82.6% accuracy rate was obtained from the validation set prediction. This metric was obtained without tuning the model. The team decided to perform cross validation using k=10 folds and combining several parameters that have been tested to find the sweet spot and generate a better accuracy if possible.

The cross validation improved the accuracy by 0.2% resulting in 82.8% and standard deviation of 0.1%. We believe different parameters tuning trials would have probably generated a better accuracy, however, 82.8% is the best we were able to achieve using a combination of several parameters. Figure 6 below summarizes the results obtained through XGBoost and cross validation.

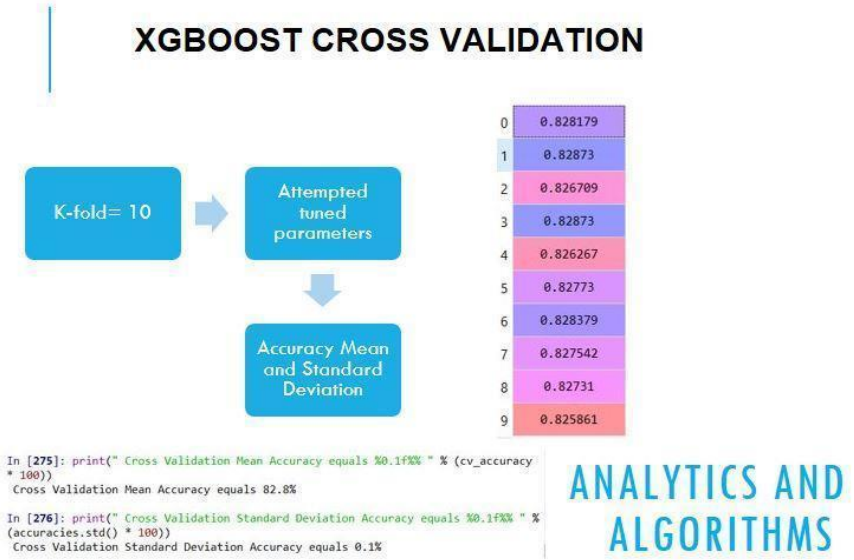
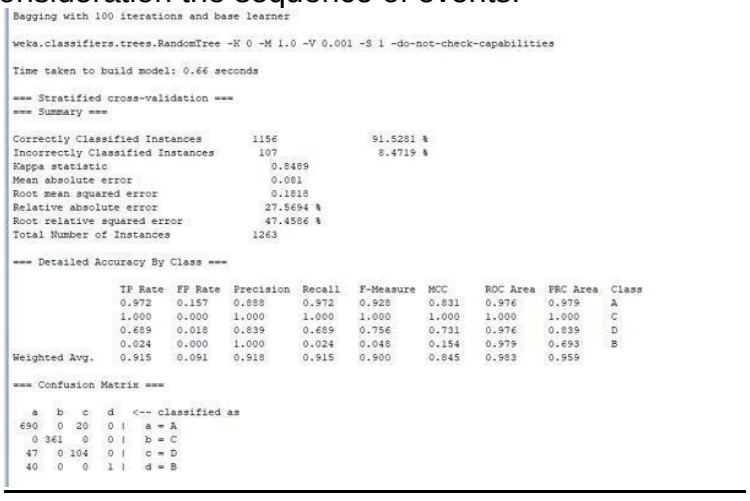


Figure 6. XGBoost Cross Validation Results

3.2.2. Random Forest

The random forest method sort to classify pilots state of mind using feature engineered predictors. In the case of this project we aggregated time into 1 sec windows and the average measurements for each feature was engineered into new features. The data wrangling was executed in MSSQL and subsequent analysis carried out in WEKA. The results as pictured below were very encouraging, however upon further research we discovered that this method did not take into consideration the sequence of events.



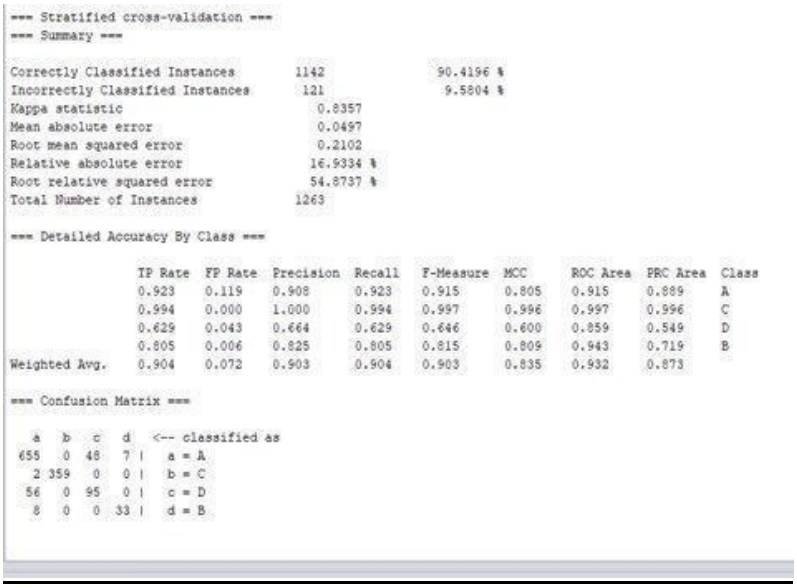
```
Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.66 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1156               91.5281 %
Incorrectly Classified Instances       107                8.4719 %
Kappa statistic                          0.8489
Mean absolute error                      0.081
Root mean squared error                  0.1818
Relative absolute error                 27.5694 %
Root relative squared error             47.4586 %
Total Number of Instances             1263

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.972     0.157     0.888     0.972     0.928      0.831     0.976     0.979     A
                 1.000     0.000     1.000     1.000     1.000      1.000     1.000     1.000     C
                 0.689     0.018     0.839     0.689     0.756      0.731     0.976     0.839     D
                 0.024     0.000     1.000     0.024     0.048      0.154     0.979     0.693     B
Weighted Avg.    0.915     0.091     0.918     0.915     0.900      0.845     0.983     0.959

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 690   0  20   0 |   a = A
   0 361   0   0 |   b = C
  47   0 104   0 |   c = D
  40   0   0   1 |   d = B
```

Figure 7. Random Forest Results

3.2.3. SVM

Using a multi-pronged approached as previously described, an SVM classifier was applied to the feature engineered dataset. This classifier technically outperformed

the random forest approach with improvements in precision and recall values for each event (see below). However just as in the random forest method this classifier also did not take into consideration the sequence of the data.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1142               90.4196 %
Incorrectly Classified Instances       121                9.5804 %
Kappa statistic                          0.8357
Mean absolute error                      0.0497
Root mean squared error                  0.2102
Relative absolute error                 16.9334 %
Root relative squared error             54.8737 %
Total Number of Instances             1263

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.923     0.119     0.908      0.923     0.915     0.805     0.915     0.889     A
               0.994     0.000     1.000      0.994     0.997     0.996     0.997     0.996     C
               0.629     0.043     0.664      0.629     0.646     0.600     0.859     0.549     D
               0.805     0.006     0.825      0.805     0.815     0.809     0.943     0.719     B
Weighted Avg.  0.904     0.072     0.903      0.904     0.903     0.835     0.932     0.873

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 655   0  48   7 |   a = A
   2 359   0   0 |   b = C
  56   0  95   0 |   c = D
   8   0   0  33 |   d = B
```

Figure 8. SVM Results

The results obtained summarized in Table 5 were somewhat decent and allowed us to predict the probability of each pilot to experiment one of CA, DA or SS cognitive states. However, although the approach provided good accuracy in predicting the event a pilot can go through, our team decided to explore the data further by trying a different approach from what most participants on the Kaggle competition seemed to have tried. The team decided to pursue a more complex machine learning technique that could deeply learn from the data's features and provide better accuracy.

Method	**Accuracy**
XGBoost	82.6% (+- 0.1%)
Random Forest	1 (max) 0.024 (min)
Support Vector Machine	1 (max) 0.624 (min)

Table 5. Summary of accuracies

3.3 . Second Approach: Long - Short Term Memory for Time Series Data

Due to the fact that all three experiments performed have been evaluated on a time window basis, it was right that a time series analysis was explored. The team decided to try this approach because all previous analysis performed were ignoring the time stamp in the data. Based on the research the team performed and advice received from subject matter experts in machine learning, long-short term memory (LSTM) has

been retained as one the best model for interpreting the features from a huge dataset across time steps.

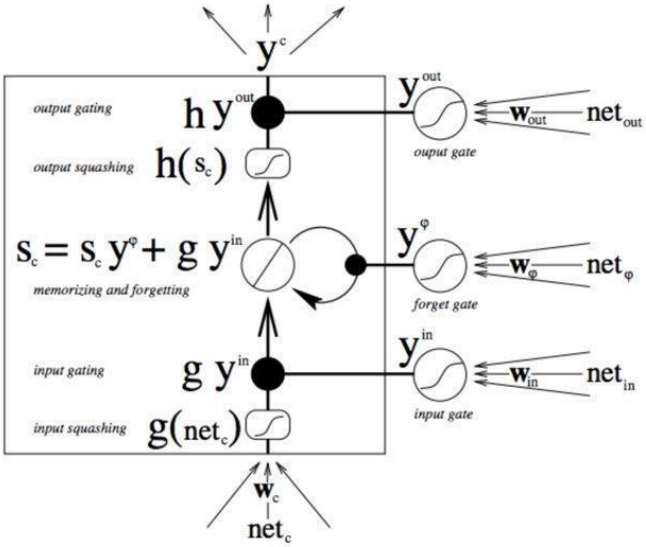


Figure 9. LSTM Diagram[18]

Why LSTM?

LSTM has been shown to provide particular good results when applied to time-series or sequential data.[19] The Long Short-Term Memory, or LSTM, network is perhaps the most successful RNN because it overcomes the problems of training a recurrent network and in turn has been used on a wide range of applications.[20] "LSTM is an artificial recurrent neural network (RNN) architecture used in the deep learning field. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series."[21]

3.3.1 . Sample Trial

The initial training dataset provided to us was split into 54 subsets. This technique was used to maintain the timestamp of our data and to better study the responses of each pilot that was subjected to an experiment. The LSTM model was then built without specification of timesteps and tested on one of the subsets to evaluate the performance of our model.

---

[18] Retrieved from https://skymind.ai/wiki/lstm

[19] Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2018). Recurrent Neural Networks for Multivariate Time Series with Missing Values

[20] Retrieved from https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

[21] Retrieved from https://en.wikipedia.org/wiki/Long_short-term_memory

The subset data of approximately 92,000 observations was split into a training set of 70% of the entire data and test set of 30%. The data was then evaluated after being trained and tested by a built 3 layers stacked LSTM model. The choice of the number of layers used in this trial approach resulted from several tests. The model was initially built using 4 layers and after fitting the model, an accuracy of 100% was obtained due to overfitting. Therefore, the number of layers had to be revised. LSTM is a data hungry algorithm, which implies that it is critical that the model be fed with the right batch size, hidden number of neurons and number of epochs to properly detect and learn the features and patterns across our data and generate good predictions. With that being mentioned, numerous iterations and implementations with different parameters to obtain a more reasonable result have been deployed. The number of epochs initially tested on 40 was considerably reduced to 2 for that specific dataset, the number of hidden layers dropped to 3, the number of hidden neurons and the batch size were altered multiple times and the activation parameter changed from softmax to sigmoid. The results from all these runs allowed the team to obtain an accuracy of about 87%.

From figure 10 below, we observe that as the model trains epoch after epoch, the cross-entropy ('loss') is minimized in both training and validation splits.

Accuracy, inversely proportional to cross-entropy, increases after the first epoch at around 85% and 87% in training and validation splits, respectively. The outcome was an accuracy of 87%, which resulted in a better prediction accuracy than what was obtained with XGBoost. We would expect a higher prediction from performing a cross validation with tuned hyper parameters.

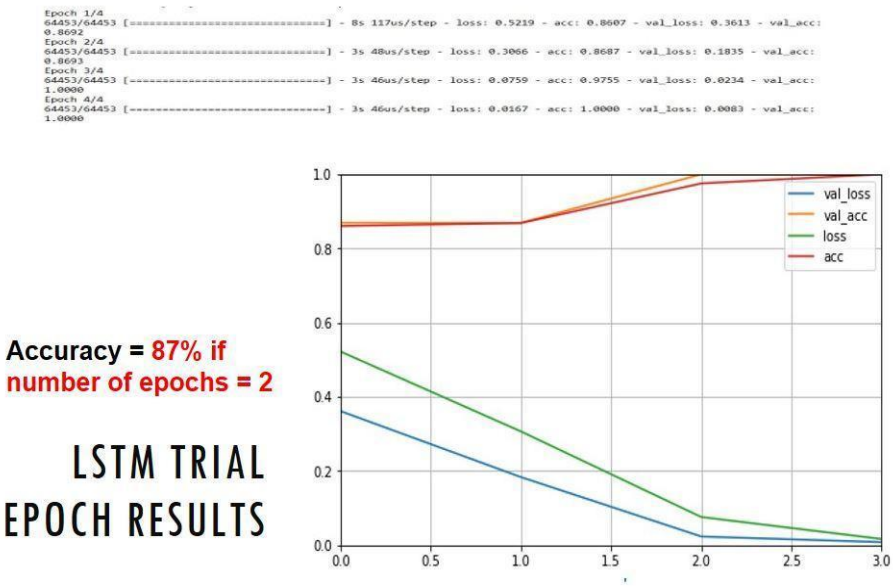


Figure 10. LSTM Trial Results

3.3.2 . Full Trial

3.3.2.1. Trial with specific sequences

Initially, the model was built following a sample sequence approach. This was achieved by creating a sequence of 35 seconds worth of records-equivalent to a size of 8960 observations by sequence- that generated an output of one event/state of mind per sequence. The goal of this approach was to teach the model how to learn from the pattern in our dataset. However, that approach had large memory demands due to the fact that it was storing all the data in memory. Even with access to high computational power of approximately 1TB offered to our team by the Operation Research Center of George Mason University (GMU), the memory was still insufficient. The team therefore had to try another approach.

3.3.2.2. Trial without sequences

We then decided to change our approach and train our model on 18 minutes (subjective) worth of records, test it on the remaining time of our dataset to predict the next timestep - next event at the nanosecond. A stacked 4 layers LSTM model has then been applied to the dataset containing records for 3 crews. The choice of the number of layers for our model was derived from readings and research. In fact, it was recommended to consider using between 2 to 5 hidden layers when building a LSTM model and dealing with large datasets to make it robust enough. The team opted to go with 4 layers. Our model has since then been in evaluation mode as parameters tuning was necessary to have the best results and beat the previous accuracies scores. As mentioned earlier, tuning an LSTM model requires numerous iterations to determine which parameters produce the best results. Our approach to run the model without pre-defining sequences has been implemented and numerous parameters tuning have been attempted. Although several runs have been submitted to the cluster, unfortunately, the evaluations have been unsuccessful due to time and computing resources constraints. The team was hoping to obtain the accuracy and loss of both the training and validation sets, perform cross validations to improve the model, visualize and predict the probabilities of each ID in the Kaggle Test data and finally as opposed to the approach followed by all teams involved in the Kaggle's competition, a time series analysis would have been plotted. This dynamic time series visualization plot expected to be generated would have allowed the team to capture the time window preceding a change in state of mind of the pilots and therefore be displayed on our prototype alerting system designed to inform pilots of their current state of minds while flying the planes.
The script architecture presented in Figure 11 provides an insight to the most recent process followed to build our LSTM model:

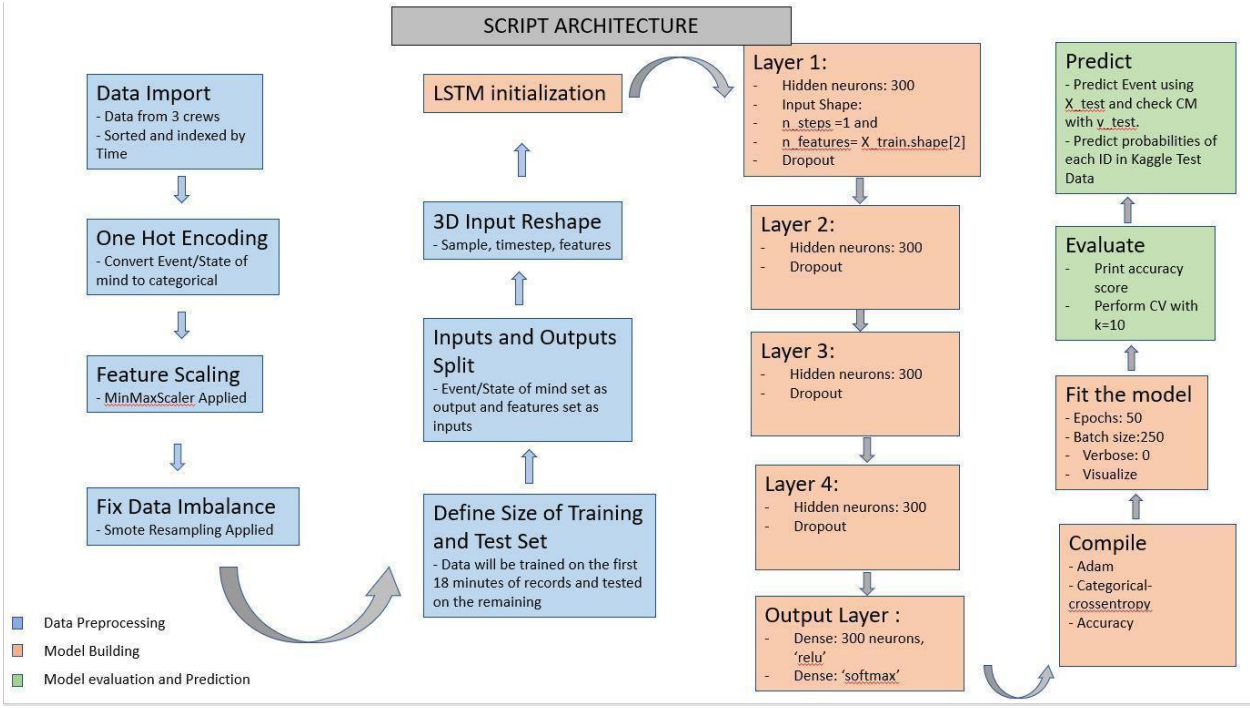


Figure 11. LSTM Script Diagram

4. Visualization

The following subsection includes visualizations developed at different stages of the project that have helped us understand our data, determine paths to take in our analysis, and to draw insights.

4.1 Training Data

4.1.2 Distribution of events labeled

Figure 12 shows the general distribution of events in the training dataset. Looking at the distribution of records included in the dataset, we understood the challenges that we would face when aggregating the data as there is a strong imbalance present.

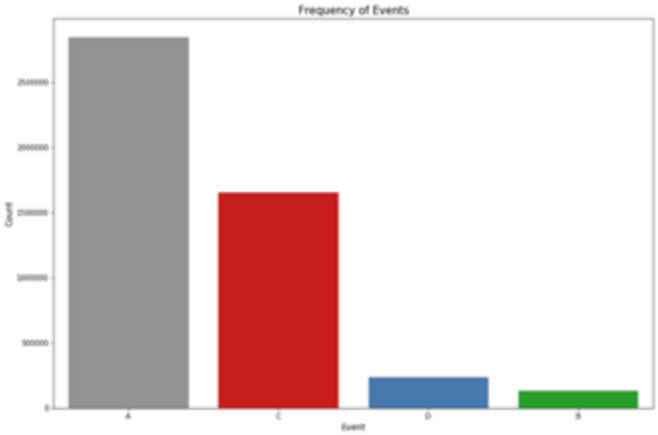


Figure 12. Events distribution

4.1.3 Baseline readings and experiment comparison

In order to gain a clear understanding of the experiments to which the crews were subjected to and to identify potential issues with the data, we decided to visualize the distribution of records in this dataset. Figure 13 shows a comparison of the distribution of records in the training dataset prior removal of outliers. From this bar plot, we have learned that the Channelized Attention experiment had a small proportion of readings labeled as baseline as compared to those labeled as channelized attention. Conversely, the Diverted Attention and Startled/Surprised experiments have longer periods in baseline. These observations make sense when we consider the nature of each of these experiments.

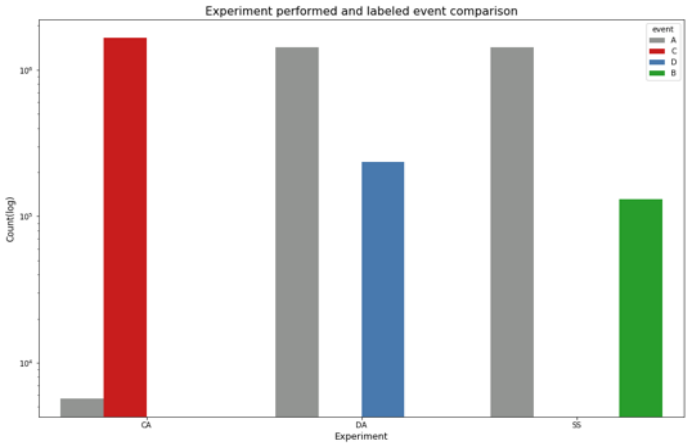


Figure 13. Baseline readings distribution across experiments

4.1.4 Influence dependent of seat assigned

The bar plot Figure 14 assesses if the pilot seating in the right was put through roughly the same duration experiments as the pilot seating on the left. The graph shows no major variation and confirms the team has no reason to explore an imbalance in this aspect further.

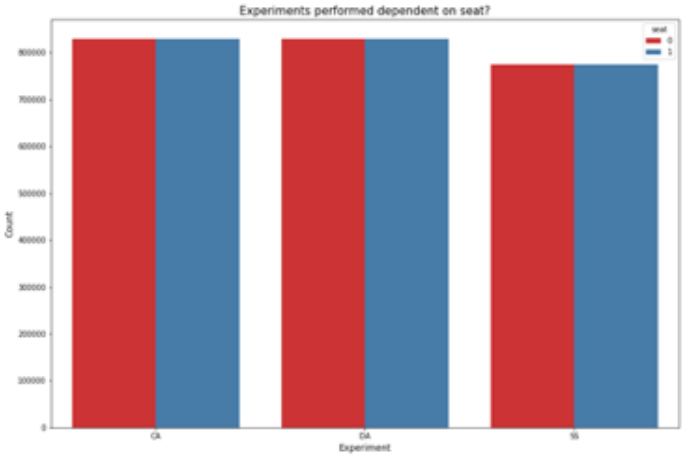


Figure 14. Pilot and co-pilot subjected to same experiments

4.2 Noise Reduction

The original physiological data collected contained peaks that exposed the presence of noise in the data. A common occurrence for physiological data collected in the manner eeg data was collected.

These were recognized as outliers as shown in the visuals below, the unusual peaks are the outliers. So, it was important to get rid of the outliers to get distinct readings for our analysis. This was done by removing values of EEG that were four standard deviations away from the mean of the values, this was done in Python using the NumPy module and its standard deviation and mean features. As we compare both graphs, we can see the difference in the range of values for the X-axis before cleaning (Figure 15) the range was almost -2500to +2500 and after cleaning (Figure 16) it was reduced to +250 and -250. Figure 16 shows the visual of the cleaned data which was then used to train our deep learning model on.

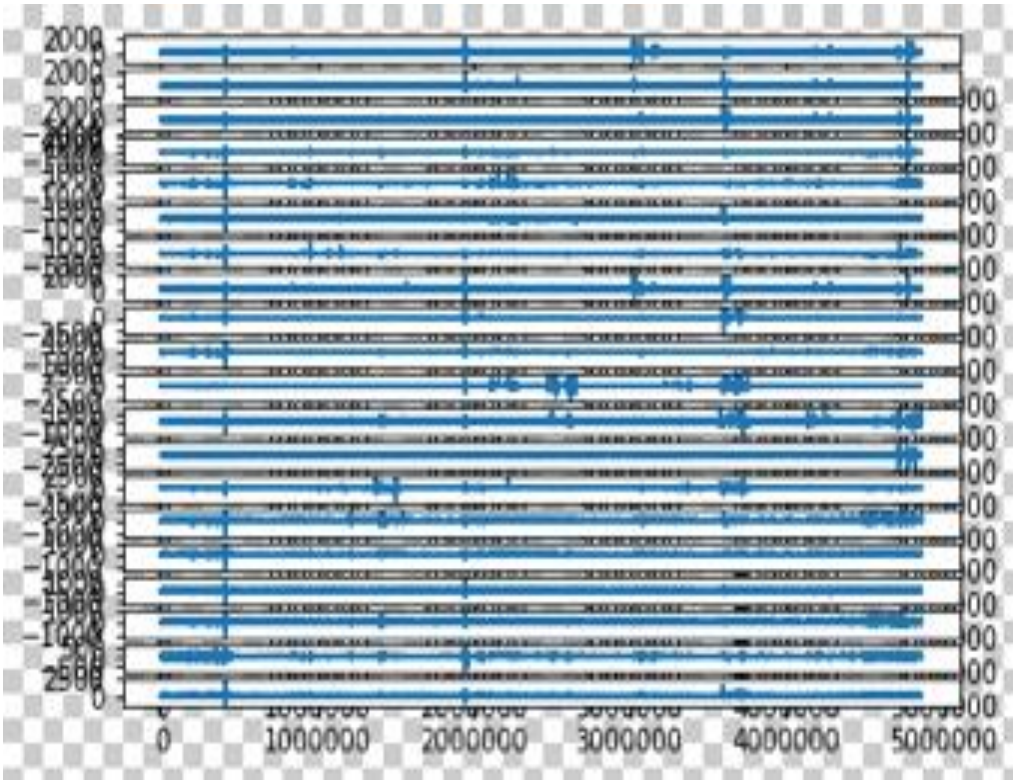


Figure 15. EEG Data With Outliers

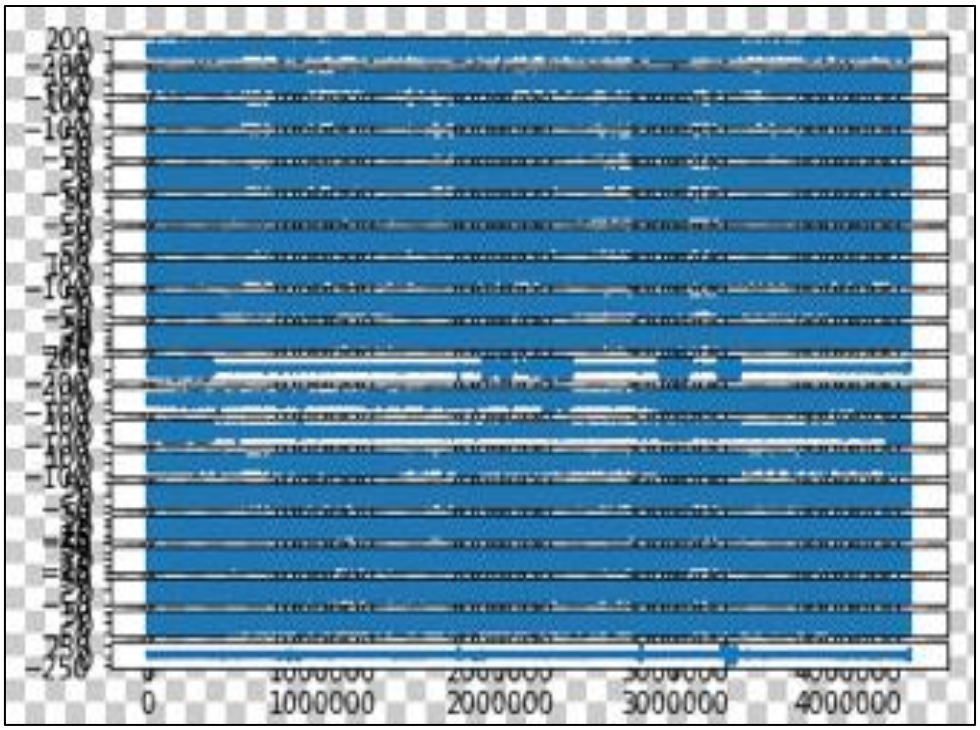


Figure 16. EEG Data Without Outliers

4.3 Team Innovation Proposal (Alerting System)

With Realtime physiological readings and state of mind predictions the team will develop an alerting system. The alerting system displays the pilots' basic information, flight details and aggregated real time physiological data. In addition to that the alerting system with a predefined threshold would alert the pilots of their undesired state of mind with visual and possibly audio cues.

4.3.1 Development Process

The idea of an alerting system was borne out of one of the team's initial brainstorming sessions. The team was initially focusing on understanding cleaning and mining the dataset. After the team had make headway with the initial focus the question of 'How to present the results?' arouse. An alerting system was initially not within the scope of the project; however, the team understands that the value of the predictions we aim at attaining, was actually in the delivery and possibly averting tragedy.

Armed with this new requirement a developer was assigned to the task of developing a visually stimulating dashboard to present results. The team took advantage of the Tableau software available to GMU students. This task was to run concurrently with the development of the predictive algorithm. Since the alerting system and algorithm development run concurrently the team decided to use a sample of the training data to develop the alerting system since the final alerting system would run on the same data structure as the training data (i.e EEG, ECG, Respiratory rate and time) to predict and outcome (EVENT). This

would allow the team to test the functionality and simulate the performance of the alerting system.

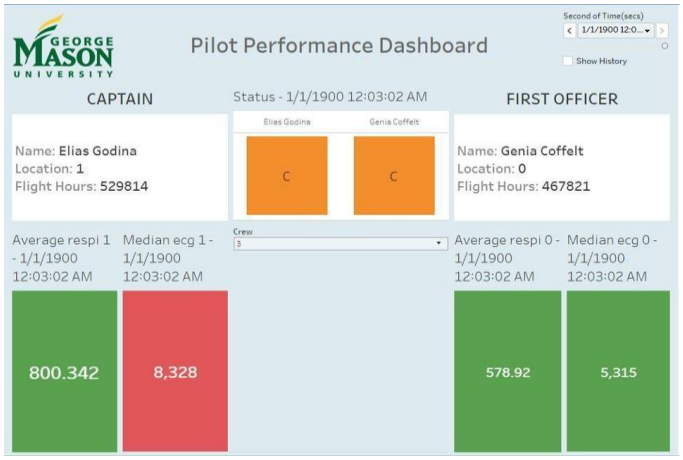


Figure 17. Initial Dashboards (Alert 1.0)

4.3.2. Final Alerting System

The dashboard displays pilot and flight information. A predefined threshold was chosen for respiratory rate and ecg reading for each pilot within a predefined time window (1 sec). The real-time respiratory rate and ecg reading are aggregated and compared to the threshold, visual cues are created for exceeding the thresholds. The centerpiece of the alerting system is the 'STATE OF MIND' predictions. For a predefined time window, the predicted state of mind is boldly displayed with text and distinct color codes (red for undesired states green for baseline). A copy of the final dashboard is available at https://public.tableau.com/views/PilotPerformanceDashboard2_0/Pilots1basicinfo?:embed=y&:display_count=yes

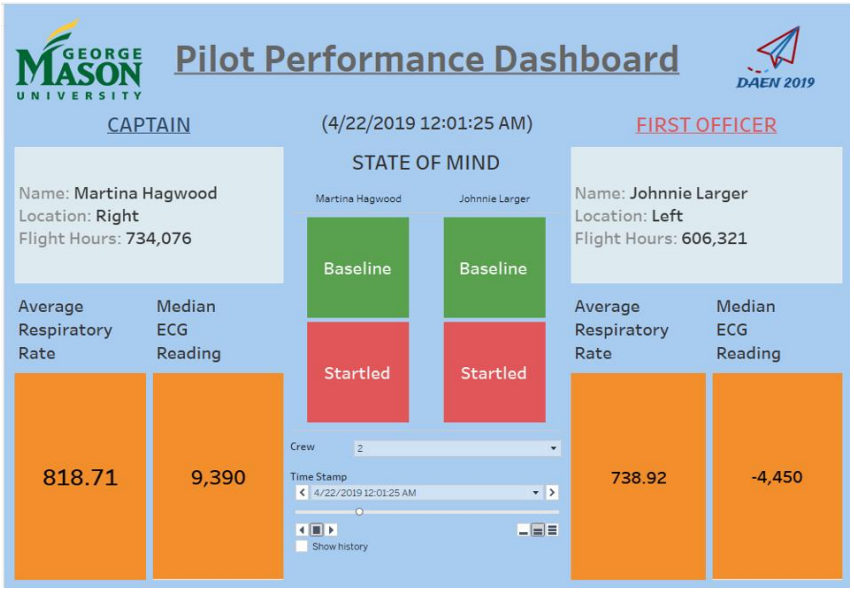


Figure 18. Final Dashboard (Alert 2.0)

5. Findings

A lot of factors contribute to the state of mind of the pilot, one of the prime factors being the seat position. Since, the main captain would have to deal with more responsibilities than the co-pilot thus keeping him (the main captain) more alert all the time. The pattern that his responses follow would be different than that of the co-pilot. The goal was to find these patterns and predict the pilot state of mind to alert them if they get distracted.

Although the classification machine learning analysis applied provided relatively good results, the LSTM trial analysis yielded a better accuracy. The team's goal was to aim for a prediction of at least 90% as it is really important to develop a robust model that could make an impact and be used to help reduce commercial aviation fatalities. However, because the approach was a trial, the team was unable to make conclusive assumptions as to the final accuracy of our model.

Our 4 layers stacked LSTM model has been built and implemented but as previously mentioned due to computing power and time constraints, the team was unable to discover the essential hyperparameters needed for the model to be predictive, proscriptive sufficient to exercise and evaluate the full model to actionable result and therefore meet the team's prediction goal. The model needed to be specifically optimized and tuned as well as the Time-series sophistication and precision. These optimizations would have been feasible by performing several iterations and runs using a high computing power.

The team believes that our script architecture provides a roadmap for the follow-on work required to prepare an optimized code and process essential for efficient computing of the necessary data to results. Thus, implying the full model will be a strong tool in the next step of meeting the project's research and demonstration objectives.

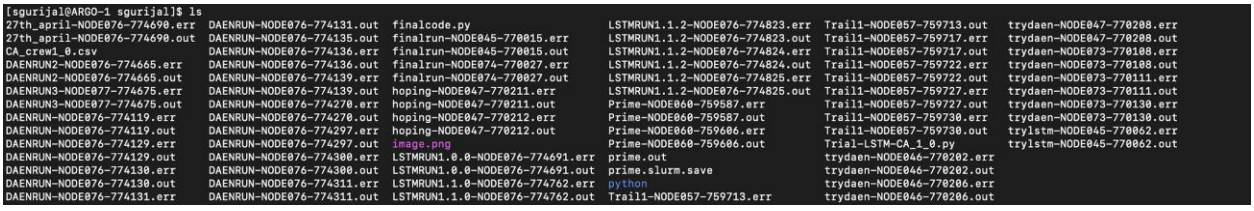


Figure 19: Outputs of various LSTM runs on Argo Cluster

Like discussed earlier in the section we have run many attempts and versions of LSTM on Argo cluster which is been provided by office of research computing. The above image shows the various outputs as .out files which are been saved in scratch folder in the cluster.

6. Summary

- The accuracies of the different machine learning models like XGBoost, Random Forest and SVM have been compared.
- Initially Exploratory Data Analysis was performed on the data to understand the data better.

- LSTM was a better performing model as compared to XGBoost.
- PCA was applied for dimensionality reduction but we ended up ignoring the time stamp, thus only outliers were removed as a part of data pre-processing.
- A simulation model was developed in Tableau.
- LSTM takes time stamp into consideration and trains over the event state of the pilot predicting the event in the test data.

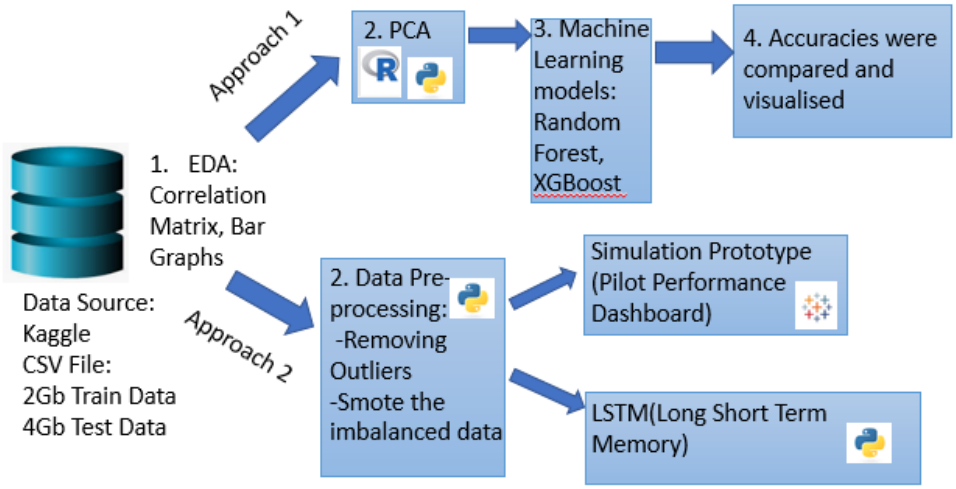


Figure 20. Methodology Summary

7. Future Work

- Check if there is possibility that the data might be stationary. If proven not stationary, consider making it stationary to improve results.
- Although a CNN-LSTM model has been developed, it has not been tested-find script in the appendix- so we strongly encourage a revised implementation of such model with different hyper parameters to find the sweet spot that will deliver the most accuracy.
- The team proposes the use of parallel processing across a cluster using Apache Spark for faster processing.
- Design and execute an experiment using simulators to measure the efficiency of the alerting system. The experiment must create as similar conditions as possible and have on alerting system for one scenario and none in the other. Consider having pilots of the same level of experience for accurate comparison and subject them to supervising, non-engaging activities.

Appendix A-Script PCA-XGBoost

```
#1) SHEBANG AND PACKAGES
#!/usr/bin/env python3
#Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
#Import Train and Test datasets
train = pd.read_csv('train.csv')
Test = pd.read_csv('test.csv')
#2) DATA STRUCTURES
#Naming selected features
Feature_eeg = ["eeg_fp1", "eeg_f7", "eeg_f8", "eeg_t4", "eeg_t6", "eeg_t5",
        "eeg_t3", "eeg_fp2", "eeg_o1", "eeg_p3", "eeg_pz", "eeg_f3",
        "eeg_fz", "eeg_f4", "eeg_c4", "eeg_p4", "eeg_poz", "eeg_c3",
        "eeg_cz", "eeg_o2"]
Feature_ecg =["ecg"]
```

```
Feature_r = ["r"]
Feature_gsr = ["gsr"]
Feature_ERG = ["ecg", "r", "gsr"]
Feature_some = ['time',"eeg_fp1", "eeg_f7", "eeg_f8", "eeg_t4", "eeg_t6", "eeg_t5",
        "eeg_t3", "eeg_fp2", "eeg_o1", "eeg_p3", "eeg_pz", "eeg_f3",
        "eeg_fz", "eeg_f4", "eeg_c4", "eeg_p4", "eeg_poz", "eeg_c3",
        "eeg_cz", "eeg_o2", "ecg", "r", "gsr"]
Feature_all =["crew", "seat", "time","experiment","eeg_fp1", "eeg_f7", "eeg_f8",
        "eeg_t4", "eeg_t6", "eeg_t5",
        "eeg_t3", "eeg_fp2", "eeg_o1", "eeg_p3", "eeg_pz", "eeg_f3",
        "eeg_fz", "eeg_f4", "eeg_c4", "eeg_p4", "eeg_poz", "eeg_c3",
        "eeg_cz", "eeg_o2", "ecg", "r", "gsr"]

# WRITTEN FUNCTIONS
#Encode categorical data
# A = baseline, B = SS, C = CA, D = DA
encode_exp = {'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': 4}
encode_event = {'A': 3, 'B': 2, 'C': 0, 'D': 1}
labels_exp = {v: k for k, v in encode_exp.items()}
labels_event = {v: k for k, v in encode_event.items()}
train['event'] = train['event'].apply(lambda x: encode_event[x])
train['experiment'] = train['experiment'].apply(lambda x: encode_exp[x])
Test['experiment'] = Test['experiment'].apply(lambda x: encode_exp[x])
train['event'] = train['event'].astype('int8')
train['experiment'] = train['experiment'].astype('int8')
Test['experiment'] = Test['experiment'].astype('int8')
#View the data
train[Feature_all].head()
#PYTHON MAIN FUNCTIONS
#Sampling Train and Test Data
```

```
Test_s = Test.sample(frac=0.1, replace=True, random_state=0)
Train = train.sample(frac=0.4, replace=True, random_state=0)
#Separate Target Variables from Train
Y = Train['event']
#Separate ID from Test
Id = Test_s['id']
#Feature Scaling
scaler = StandardScaler()
Train[Feature_all] = scaler.fit_transform(Train[Feature_all])
Test_s[Feature_all] = scaler.transform(Test_s[Feature_all])

#Apply PCA
pca = PCA(.90)
pca.fit(Train[Feature_all])
pc=pca.n_components_
print(pc)
TrainFinal = pca.transform(Train[Feature_all])
TestFinal = pca.transform(Test_s[Feature_all])
Variance = pca.explained_variance_ratio_
print(Variance)
#Fix imbalance in event
TrainFinal, Y = SMOTE().fit_resample(TrainFinal, Y.ravel())
#Splitting the data into the training set and the test set
X_Train, X_Test, Y_Train, Y_Test = train_test_split(TrainFinal, Y, test_size = 0.3,
                                    random_state = 0)
# Create an instance of the chosen classifier.
params = {
        'objective': 'multiclass',  # error evaluation for multiclass training
        'num_class': 4,
        "metric" : "multi_error",
```

```
        "learning_rate" : 0.1,
        "min_child_weight" : 40,
        "feature_fraction" : 0.8,
        "reg_alpha": 0.15,
        "reg_lambda": 0.15,
        'n_gpus': 0}
classifier = XGBClassifier(objective= 'multiclass',
        num_class= 4,
        metric = "multi_error",
        learning_rate = 0.1,
        min_child_weight = 40,
        feature_fraction = 0.8,
        reg_alpha= 0.15,
        reg_lambda= 0.15)
n_classifier = len(classifier)
#Apply Xgboost Classifier and print accuracy score
for index, (name, classifier) in enumerate(classifier.items()):
        classifier.fit(X_Train, Y_Train)

Y_pred = classifier.predict(X_Test)
Y_pred = pd.DataFrame(Y_pred)
Y_pred.describe()
accuracy = accuracy_score(Y_Test, Y_pred)
print("Accuracy (Train) for %s: %0.1f%% " % (name, accuracy * 100))

# Probabilities:
probas = classifier.predict_proba(X_Test)
probas
# Making the Confusion Matrix
cm = confusion_matrix(Y_Test, Y_pred)
```

```
print(cm)
# Applying k-Fold Cross Validation with Parameter Tuning
accuracies = cross_val_score(estimator = classifier, X = X_Train, y = Y_Train,
                             cv = 10, verbose=0)
cv_accuracy=accuracies.mean()
print(" Cross Validation Mean Accuracy equals %0.1f%%  " % (cv_accuracy * 100))
print(" Cross Validation Standard Deviation Accuracy equals %0.1f%% " %
(accuracies.std() * 100))


#Plot Confusion Matrix result
def plot_confusion_matrix(cm, classes, normalized=True, cmap='bone'):
        plt.figure(figsize=[7, 8])
        norm_cm = cm
        if normalized:
        norm_cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        sns.heatmap(norm_cm, annot=cm, fmt='g', xticklabels=classes,
yticklabels=classes, cmap=cmap)
        plt.savefig('confusion-matrix.png')
plot_confusion_matrix(cm, ['C', 'D', 'B', 'A'])
#Predict probability of ID's in Test Data
Y_pred1 = classifier.predict(TestFinal.values)
Y_pred1= classifier.predict_proba(TestFinal)
proba = Y_pred1
print(Y_pred1)
#Export Prediction
sub = pd.DataFrame(Y_pred1, columns=['A', 'B', 'C', 'D'])
sub['id'] = Id
cols = sub.columns.tolist()
cols = cols[-1:] + cols[:-1]
sub = sub[cols]
```

```
sub.to_csv("Test_prob.csv", index=False)
```

Appendix B- Script Trial LSTM

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 17:25:46 2019
@author: Koffi Paule Carelle
"""
# I- Data Preprocessing
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
Dataset = pd.read_csv('crew1_leftSeat_DA.csv')
Dataset.drop(['Unnamed: 0'], axis=1, inplace=True)
Dataset['time'] = Dataset['time'].astype('timedelta64[s]')
Dataset.info()
Dataset.head()
Dataset.to_csv('air pilot.csv')
#Data selection
dataset = pd.read_csv('air pilot.csv', index_col='time')
dataset.drop(['Unnamed: 0'], axis=1, inplace=True)
dataset.index.name = 'time'
#Quick visualization some of the parameters
Eeg_f1 =1 np.array([Dataset.iloc[:,4]])
Ecg = np.array([Dataset.iloc[:,24]])
R = np.array([Dataset.iloc[:,25]])
GSR= np.array([Dataset.iloc[:,26]])
```

```
plt.figure(1)
Eeg_f1, = plt.plot(Eeg_f1[0,:])
Ecg, = plt.plot(Ecg[0,:])
R, = plt.plot(R[0,:])
GSR, = plt.plot(GSR[0,:])
plt.legend([Eeg_f1,Ecg,R,GSR], ["Eeg_f1","Ecg","R", "GSR"] )
plt.show()
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder
values = dataset.values
labelencoder_1 = LabelEncoder()
values[:, 1] = labelencoder_1.fit_transform(values[:, 1])
labelencoder_2 = LabelEncoder()
values[:, 26] = labelencoder_2.fit_transform(values[:, 26])
values = values.astype('float32')
# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
values_scaled = scaler.fit_transform(values)
# Drop columns we don't want to predict
Final= pd.DataFrame(values_scaled)
Final.drop([0], axis=1, inplace=True)
Final.drop([1], axis=1, inplace=True)
Final.drop([2], axis=1, inplace=True)
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X = Final.iloc[:, 3:26].values
y = Final.iloc[:, 23].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
# Reshaping
```

```
X_train = np.reshape(X_train, (X_train.shape[0],1, X_train.shape[1]))
X_test = np.reshape(X_test,(X_test.shape[0], 1, X_test.shape[1]))
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
#II - Building the RNN
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
# Initialising LSTM
classifier = Sequential()
# Adding the first LSTM layer and some Dropout regularisation
classifier.add(LSTM(units = 100, return_sequences = True,
                    input_shape = (X_train.shape[1], X_train.shape[2])))
classifier.add(Dropout(0.3))
# Adding a second LSTM layer and some Dropout regularisation
classifier.add(LSTM(units = 100, return_sequences = True))
classifier.add(Dropout(0.4))
# Adding a third LSTM layer and some Dropout regularisation
classifier.add(LSTM(units = 100))
classifier.add(Dropout(0.5))
# Adding a fourth LSTM layer and some Dropout regularisation
#classifier.add(LSTM(units = 100))
#classifier.add(Dropout(0.2))
# Adding the output layer
classifier.add(Dense(units = 1, activation='sigmoid'))
# Compiling LSTM
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])
print(classifier.summary())
```

```
# Fitting the RNN to the Training set
history = classifier.fit(X_train, y_train, epochs = 4, batch_size = 1500,
                         validation_data=(X_test, y_test))
# Final evaluation of the model
scores = classifier.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
# summarize scores
from numpy import mean
from numpy import std
def summarize_results(score):
 print(score)
 m, s = mean(score), std(score)
 print('Accuracy: %.3f%% (+/-%.3f)' % (m, s))

# run an experiment
def run_experiment(repeats=10):
 # repeat experiment
 score = list()
 for r in range(repeats):
  score = scores(X_train, X_test, y_train, y_test)
  score = score * 100.0
  print('>#%d: %.3f' % (r+1, score))
  scores.append(score)
# summarize results
          summarize_results(score)
# III- Making the predictions and visualising the results
```

```
# Getting prediction for test set
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
y_pred1= classifier.predict_proba(X_test)
# invert scaling for forecast
from numpy import concatenate
X_test = X_test.reshape((X_test.shape[0], X_test.shape[2]))
inv_ypred = concatenate((y_pred, X_test[:, 1:]), axis=1)
inv_ypred = scaler.inverse_transform(inv_ypred)
inv_ypred = inv_ypred[:,0]
# invert scaling for actual
y_test = y_test.reshape((len(y_test), 1))
inv_y = concatenate((y_test, X_test[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
# Visualize results
plt.figure()
plt.plot(y_test,y_pred)
plt.show(block=False)
plt.figure()
test, = plt.plot(y)
predict, = plt.plot(y_pred)
plt.legend([predict,test],["predicted Data","Real Data"])
plt.show()
```

Appendix C- Script LSTM Final Approach

```
# -*- coding: utf-8 -*
```

```
#!/usr/bin/env python3-
"""
Created on Tue Apr  2 19:49:29 2019
@author: Koffi Paule Carelle
"""
# I- Data Preprocessing
# Importing the libraries
#import matplotlib.pyplot as plt
#from matplotlib import pyplot
#from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
from numpy import array
from numpy import hstack
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from numpy import mean
from numpy import std
from numpy import concatenate
from sklearn.metrics import confusion_matrix

#Data selection
train = pd.read_csv('Train_new.csv', index_col='time')
train.index.name = 'time'
```

```
train.sort_index(inplace=True)
train.drop(['Unnamed: 0','crew','experiment'], axis=1, inplace=True)
# Encoding categorical data
values = train.values
labelencoder = LabelEncoder()
values[:, 24] = labelencoder.fit_transform(values[:, 24])
values = values.astype('float32')
# Feature Scaling
scaler = MinMaxScaler(feature_range=(0, 1))
values[:, 0:24] = scaler.fit_transform(values[:, 0:24])
#Define X and Y
XFinal= values[:, 0:24]
Y= output = values[:, 24]
#Fix imbalance in event
XFinal, Y = SMOTE().fit_resample(XFinal, Y.ravel())
#Set inputs and outputs
inputs = XFinal.reshape((len(XFinal), 24))
output = Y.reshape((len(Y), 1))
# horizontally stack columns
dataset = hstack((inputs, output))
# split into train and test sets
n_train_minutes = 360*3*256 #train on first 18 minutes
train = dataset[:n_train_minutes, :]
test = dataset[n_train_minutes:, :]
# split into input and outputs
X_train, y_train = train[:, :-1], train[:, -1]
X_test, y_test = test[:, :-1], test[:, -1]
# reshape input to be 3D [samples, timesteps, features]
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
```

```
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
# one hot encode y
X_train = to_categorical(X_train)
y_test = to_categorical(y_test)
# Initialising the model
classifier = Sequential()
# Adding the first LSTM layer and some Dropout regularisation
verbose, epochs, batch_size = 0, 50, 250
n_steps, n_features, n_outputs = X_train.shape[1], X_train.shape[2], y_train
classifier.add(LSTM(units = 300, return_sequences = True,
                    input_shape = (n_steps, n_features)))
classifier.add(Dropout(0.3))
# Adding a second LSTM layer and some Dropout regularisation
classifier.add(LSTM(units = 300, return_sequences = True))
classifier.add(Dropout(0.3))
# Adding a third LSTM layer and some Dropout regularisation
classifier.add(LSTM(units = 300, return_sequences = True))
classifier.add(Dropout(0.3))
# Adding a fourth LSTM layer and some Dropout regularisation
classifier.add(LSTM(units = 300))
classifier.add(Dropout(0.3))
# Adding the output layer
classifier.add(Dense(units = 300, activation='relu'))
classifier.add(Dense(1, activation='softmax'))
# Compiling the model
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
classifier.summary()


# Fitting the model to the Training set
```

```
history = classifier.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
                         verbose=verbose, validation_data=(X_test, y_test))
# Final evaluation of the model
accuracy = classifier.evaluate(X_test, y_test, batch_size=batch_size, verbose=0)
print("Accuracy: %.2f%%" % (accuracy[1]*100))
# plot history for loss and accuracy of the model
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
#2nd history plot visualization
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize scores
```

```
def summarize_results(scores):
 print(scores)
 m, s = mean(scores), std(scores)
 print('Accuracy: %.3f%% (+/-%.3f)' % (m, s))

# run an experiment
def run_experiment(repeats=10):
 # repeat experiment
 scores = list()
 for r in range(repeats):
  score = accuracy(X_train, X_test, y_train, y_test)
  score = score * 100.0
  print('>#%d: %.3f' % (r+1, score))
  scores.append(score)
 # summarize results
 summarize_results(scores)
# III- Making the predictions
# Getting prediction for test set
y_pred = classifier.predict(X_test)
print(y_pred)
# invert scaling for forecast
X_test = X_test.reshape((X_test.shape[0], X_test.shape[2]))
inv_ypred = concatenate((y_pred, X_test[:, 1:]), axis=1)
inv_ypred = scaler.inverse_transform(inv_ypred)
inv_ypred = inv_ypred[:,0]
# invert scaling for actual
y_test = y_test.reshape((len(y_test), 1))
inv_y = concatenate((y_test, X_test[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
```

```
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
#Plot the prediction
pyplot.plot(inv_ypred)
pyplot.plot(inv_y)
pyplot.show()
#IV-Prediction on real Test Data
#load test data
test = pd.read_csv('test.csv', index_col='time')
test.drop(['Unnamed: 0'], axis=1, inplace=True)
test.index.name = 'time'
test_id = test['id']
test.drop(['id', 'crew', 'experiment'], axis=1, inplace=True)

# Feature Scaling
values_test = test.values
scaler = MinMaxScaler(feature_range=(0, 1))
values_test[:,0:2] = scaler.fit_transform(values[:,0:24])
#Predict probabilities of Ids in Test data
Test= pd.DataFrame(values)
pred = classifier.predict_proba(Test)

sub = pd.DataFrame(pred,columns=['A', 'B', 'C', 'D'])
sub['id'] = test_id
cols = sub.columns.tolist()
cols = cols[-1:] + cols[:-1]
sub = sub[cols]
sub.to_csv("Test_prob.csv", index=False)
```

Appendix D- Script to remove outliers

```
# visualize dataset
import pandas as pd
from pandas import read_csv
from matplotlib import pyplot
# load the dataset
data = read_csv('C:/Users/HP/Desktop/train.csv', header=None,skiprows = 1,
usecols=range(4,24))
# retrieve data as numpy array
values = data.values
for i in range(values.shape[1]):
    pyplot.subplot(values.shape[1], 1, i+1)
    pyplot.plot(values[:, i])
pyplot.show()
# remove outliers from the EEG data
from pandas import read_csv
from numpy import mean
from numpy import std
from numpy import delete
from numpy import savetxt
# step over each EEG column
for i in range(values.shape[1] - 1):
# calculate column mean and standard deviation
    data_mean, data_std = mean(values[:,i]), std(values[:,i])
# define outlier bounds
    cut_off = data_std * 4
    lower, upper = data_mean - cut_off, data_mean + cut_off
# remove too small
    too_small = [j for j in range(values.shape[0]) if values[j,i] < lower]
    values = delete(values, too_small, 0)
```

```
    print('>deleted %d rows' % len(too_small))
    # remove too large
    too_large = [j for j in range(values.shape[0]) if values[j,i] > upper]
    values = delete(values, too_large, 0)
    print('>deleted %d rows' % len(too_large))
# save the results to a new file
#savetxt('C:/Users/HP/Desktop/crew1_leftSeat_DA.csv', values, delimiter=',')
# create a subplot for each time series
pyplot.figure()
for i in range(values.shape[1]):
        pyplot.subplot(values.shape[1], 1, i+1)
        pyplot.plot(values[:, i])
pyplot.show()
pd.DataFrame(values).to_csv("C:/Users/HP/Desktop/train_nooutliers.csv")
```

Appendix E- Agile Development

For the management of the project we decided to apply an Agile methodology, more specifically scrum. To keep up with a collaborative development, the team agreed on the following fours ceremonies tailored to the team availability.

- Sprint Planning meeting- To be held at the beginning of each sprint. Product owner will present the team with goals (GMU's guide). During the meeting all team members will review the product backlog and agree on which stories are to be prioritized as well as the breakdown of these into tasks.

- Daily scrum- as needed. At least every two days per week. Scrum master is available to assist at any time if needed.

- Sprint Review- Showcase of sprint's achievements. Feedback from stakeholder (sponsor) expected. A two-hour review meeting is expected for a two-weeks long sprint.

- Sprint Retrospective- will not be held as official scrum ceremony. Instead the team is encouraged to bring up suggestions for improvement at any point during the project.