**Developing an AI/ML activity for a BME physiology course  
*\*\*\*Authors note: rough versions of some supplemental materials are included in the document for review. We will cloud host all materials for the final version, it would be difficult to blind them otherwise \*\*\****

# Introduction

The current employment landscape is likely to undergo significant changes as the prevalence of data-driven work increases. Both the types of engineering jobs available and the skills required for these jobs will be affected[1](https://paperpile.com/c/WJ14MC/wklx). Rather than the traditional computational skills (e.g. writing code, data structures, statistics), critical skills for graduates of engineering degree programs will shift to a higher level - including the ability to conceptualize, identify, organize, and make sense of data using statistical and machine learning tools. More importantly, how engineers use the results from data to solve engineering problems is already changing[1–3](https://paperpile.com/c/WJ14MC/wklx+9JxR+kpHn). In our undergraduate biomedical engineering (BME) program, we have taken to referring to this shift as one of “data skills” rather than Artificial Intelligence (AI) or Machine Learning (ML) specifically.

Our curriculum is actively shifting to increase the engagement of students in data skills-related learning across multiple classes and through a broad set of opportunities tailored to the needs of different career paths and the interests of different students. Many programs have created specific course(s), or majors or minors in Data Science[2,4](https://paperpile.com/c/WJ14MC/9JxR+p4to). While adding a major or minor is effective for students who choose this type of coursework, we see a broader need in creating a variety of opportunities. All students should have some exposure, alongside opportunities for greater depth that flow from this introductory exposure.

Introducing data skills in undergraduate BME education poses particular challenges. While many BME programs are structured so that students receive a broad range of coursework, this breadth of work may come at the cost of depth into topics critical for their future careers in the field[5](https://paperpile.com/c/WJ14MC/1NhO). Indeed, our department determined that developing a separate introductory course in data skills would not be feasible because of limits on degree credits, space, and other practical challenges. In addition, studies have shown that BME students have a limited view of the types of careers that they may attain after graduation[6,7](https://paperpile.com/c/WJ14MC/8Ket+LcUw) - in part attributed to the presentation of different topics as broad and disparate. Thus, we were motivated to integrate data skills content into existing required courses in ways that both furthered learning in the course itself and addressed our practical challenges. The result is a program of curricular change where data skills elements are incorporated into each required undergraduate BME course [BLINDED-FORTHCOMING].

In this paper, we describe the development, implementation, and evaluation of one such activity - using ML diagnostics of Atrial Fibrillation from electrocardiogram (ECG) data in a junior-level Systems Physiology course. We developed and tested two forms of this activity: a longer form requiring access to GPU computing services, and a shorter form able to be run on laptops that uses a pre-trained neural network. We used pre-post surveys to measure student perceptions of their skills in and career inclinations towards data skills abilities, and their perceptions of the applicability of data skills in jobs. The activity could be easily adopted in similar classes, and includes suggestions on how to scale it to range from 30 minutes to 2 hours of course time. We also report on our general approach for developing this activity so that other institutions may consider using this model for teaching data skills to their students. All files for the activity are provided in appendices.

# Activity and Implementation

The activity we developed was meant to serve both data skills and course-specific learning objectives. In this section we describe the curricular and course context as well the activity. As noted in the intro, this activity is part of a larger data skills initiative in our undergraduate curriculum. Our effort to change BME undergraduate courses exists within a rapidly increasing ecosystem of AI/ML learning opportunities for students at our university. These include a AI/ML-centric minor available to all engineering students. [DETAILS BLINDED]. Other opportunities include [BLINDED].

## Data skills in our BME curriculum

Emerging from ongoing concerns about preparing BME students for modern engineering work, the BME faculty and other stakeholders engaged in a process of discussion, ideation, and definition of what role AI/ML should play in our curriculum. The result was a set of data skills, shown below. The data skills were reviewed and approved by the department’s faculty and external advisory board and map closely to those identified by industry leaders in other research[8](https://paperpile.com/c/WJ14MC/gUbY). During development, consensus was that the rapid emergence of AI/ML represented a motivation and framing of the need for our students to develop data skills. However, a narrow focus would likely distract from what students actually need to be effective engineers. Therefore, we defined a set of general data skills, with obvious connections to AI/ML as well as connections to other ways in which data-driven engineering occurs. We see situating AI/ML as a continuation of the use of data, rather than as an entirely separate skillset, as important. The data skills approved for the undergraduate curriculum were:

1. Implement algorithms for data analysis as a working code in a programming language such as Python, MATLAB, R, or C/C++
   1. Write scripts combining off-the-shelf data processing tools
2. Describe probabilistic models and demonstrate the use of standard tools of statistical analysis and machine learning
3. Use regular operations in spreadsheet programs (e.g. Excel)
4. Use data to solve engineering problems in biology and medicine
   1. Apply statistical analysis and machine learning tools to different datasets and understand their limitations
   2. Justify design decisions, inputs, and constraints
   3. Recognize biases or underlying assumptions within datasets and that their use may pose risks to certain populations
   4. Organize and present data visually to an audience

In parallel with approving the skills themselves, the faculty approved a requirement that some aspect(s) of these data skills must be integrated into each required undergraduate course. While the scope of that integration was left open-ended, it must be transparent and identifiable to both instructors and students. The integrations were intended to generally adopt the following pedagogical principles:

* Meaningfully link the core content of a course to an authentic application of use of data, machine learning, statistics, and/or use of engineering computing tools in biomedical engineering work.
* Enhance, rather than replace or compress, learning about existing content in the course
* Not require any pre-requisite knowledge beyond that already required for the course
* Focus on application and exposure over first principles or mastery

Two foundational educational theories guide specific interventions enacting these principles. The first was the ICAP framework[9](https://paperpile.com/c/WJ14MC/WVSt). The nature of these interventions as application-based and focused on linking led to a focus on collaborative and student-directed exploration of AI/ML applications wherever possible. The second was a focus on higher level and forward-looking reflection as described by Kember and colleagues[10](https://paperpile.com/c/WJ14MC/m6GR). Such activities are already common in our program and are largely guided by established frameworks including inquiry and project-based learning, and conceptual change [11](https://paperpile.com/c/WJ14MC/mbMb).

## Course description

The course in which we implemented this activity is an upper-level introductory physiology course that consists of two 1-hour lectures and one 2-hour [*blinded- active learning session*] per week [BLINDED]. The active learning sessions emphasize evaluative and applicational questions (i.e., higher-levels of Bloom’s taxonomy[12](https://paperpile.com/c/WJ14MC/Fyi9)). They are built around medical case studies and typically ask students to make treatment or other decisions in the case studies. The primary objective of the active sessions is to help learners integrate content knowledge to develop conceptual understanding and solve novel problems in biomedical engineering (BME).

The course is designed to help learners connect disarticulated physiological concepts to solve system-wide problems. The general content covered is typical of introductory physiology courses[13,14](https://paperpile.com/c/WJ14MC/GImv+EalG). However, the course is somewhat different in the extensive focus on integrating knowledge across individual organs and systems to troubleshoot signs and symptoms affecting the entire system. For example, the session of the course our activity affected previously focused on connections between cardiac dysfunction (as noted through ECG readings) and connections to the respiratory system and respiratory symptoms.

## Activity

We designed the data skills activity to take the entirety of one active learning session in the systems physiology course[15](https://paperpile.com/c/WJ14MC/iJsl). That is, the activity is designed to be 110 minutes assuming that students come in with the appropriate software installed and configured. Students work in groups of 4 on a worksheet that guides them through the activity. During the course of the fall of 2024, we developed 2 versions of the data skills activity which we implemented with two different groups of students for evaluation. This section describes the activity development at a high level and then specific activities in each are described in two subsections below.

Starting from the existing cardiac/respiratory dysfunction activity as mentioned above, we revised the activity to integrate training and inference of machine learning models using prewritten and preprovided code. That is, no physiology content was removed, although some did change forms. For example, previously students visually identified P, Q, R, S, and T waves on a single ECG to assess whether they could identify those features. In the revised activity, students were asked to do so while theorizing features of the wave that could be translated into mathematical functions for computational purposes. Further, each group was asked to manually classify multiple ECGs randomly presented from the training data set which they entered into a shared spreadsheet. Across the many groups in each section (and across sections) the resulting data was automatically analyzed for accuracy, sensitivity, and specificity. Those values were compared to literature data on the accuracy of different levels of clinicians as well as to the accuracy of the machine learning models that they trained. After this introduction, students either trained three machine learning models on a large open data set (Version 1) or used pretrained models (Version 2) to classify a large open data set from the cardiac domain.

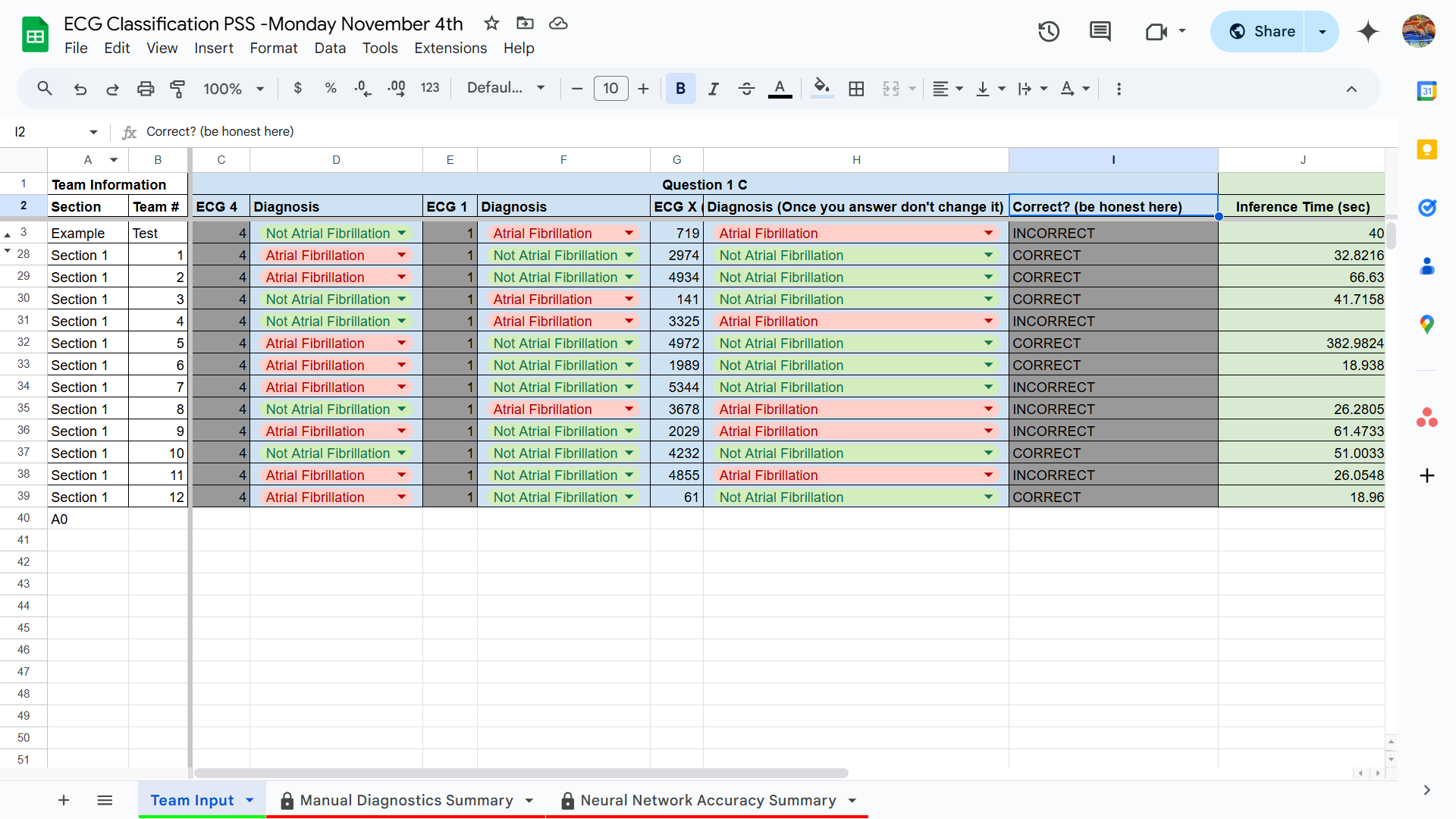
The models train neural networks to classify ECG signals as either Atrial Fibrillation (AFib) or not AFib. A classification of “not Afib” could mean anything from a completely normal ECG trace to one with ventricular fibrillation, or even an ECG with incorrect electrical connections. The training uses data from the 2017 edition of the George B. Moody PhysioNet Computing in Cardiology Challenge[16](https://paperpile.com/c/WJ14MC/e1MI). That challenge asked teams of researchers to create classification algorithms to aid clinicians in interpreting ECG signals among four possible categories.The simplified version we employed is based on example code created by MathWorks in the MATLAB software language[17](https://paperpile.com/c/WJ14MC/6gL6). That code was rewritten in several ways. First, it was refactored and organized to align with the activity prompts and questions. Second, much of the comments explaining the code itself were replaced with comments explaining the practical aspects and modeling decisions that the code reflected. This included obfuscating or glossing over aspects of the code that were extraneous to the activity. Finally, a third model training run in which groups made choices on how to modify their model to make it better, was added. The code for the activity is included in supplemental materials. As noted, we prepared two versions, the full Version 1 includes the actual training code as well as the inference code but requires significant computational resources while the compressed Version 2 skips the training but can be run efficiently on typical student laptops. We describe the compressed Version 2 first then the full Version 1.

Given common curricular change concerns in engineering education, we want to dissuade others from thinking that the use or pre-existing code is a negative for this activity, or that there is a tension between covering any specific ML model in depth and covering cardiac physiology. Our focus, as laid out in learning objectives and design principles for these activities, was engagement specifically on (1) the evaluation of the output and efficacy of such models, (2) the relationship of machine learning to the existing course content, and (3) the relationship between machine learning and clinical decision making. We saw having students write their own code as an inefficient use of limited time for such an activity in a physiology course. More importantly, the choice to use an established open data challenge and existing demonstration code was motivated by our interest in showing opportunities for continued growth to students through these activities. Doing so uncovers potential future learning and growth opportunities to students that they otherwise may not know exist. While not directly related to the intended learning outcomes, these types of choices are fundamental to our design principles for these activities. Students who find interest through these introductory exposures have multiple opportunities for future growth as described in the curricular context section.

## Compressed implementation

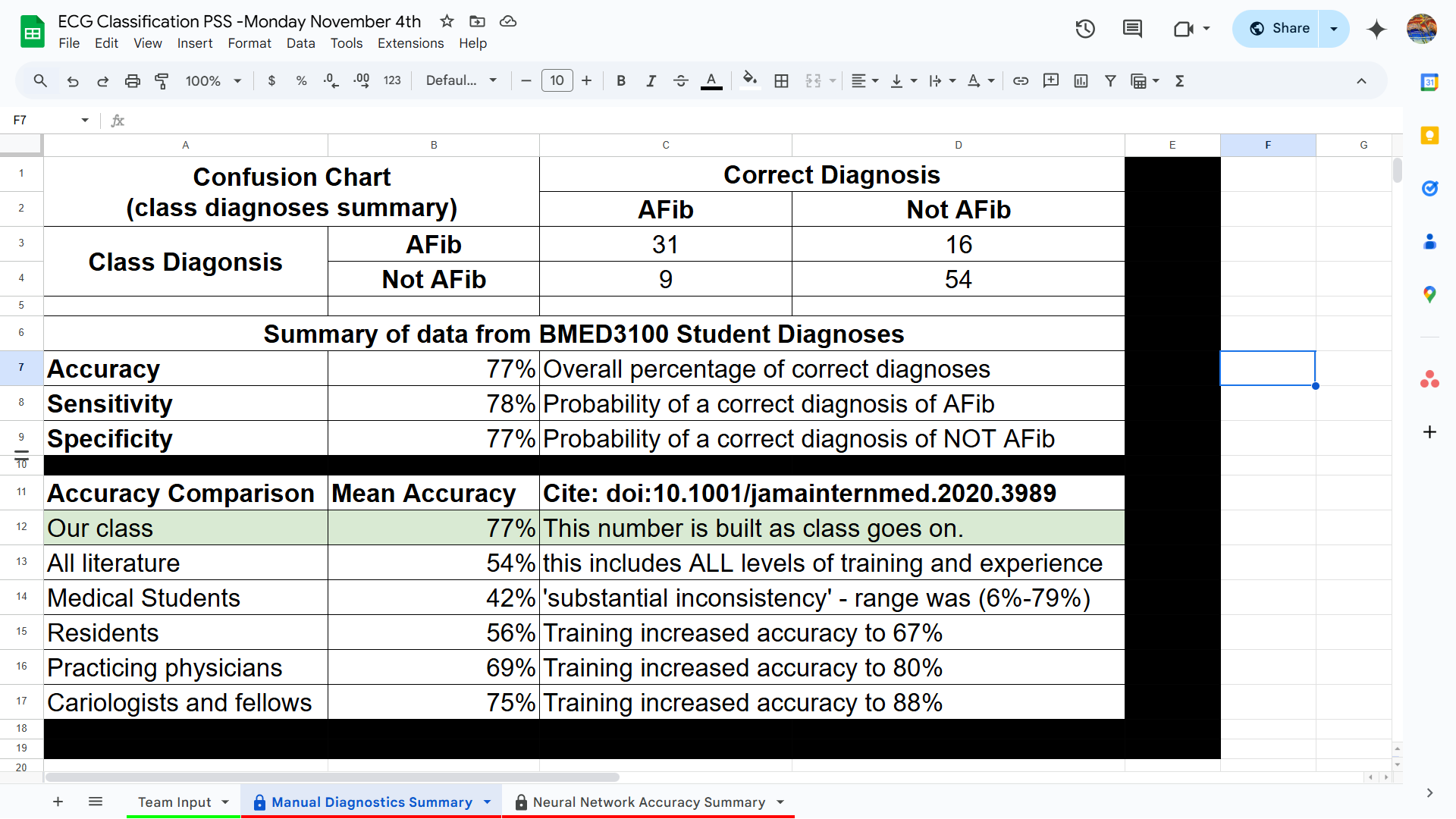
Version 2 of the activity was implemented in the first 30 minutes of a 110 minute course session, we refer to this as the compressed version of the activity. This version was implemented in the Fall of 2024 in 3 course sections, each with 48 students working in teams of 4. In this version, students rely on a pretrained machine learning model; run all code locally (i.e., on their laptop); and deal primarily with model inference, model interpretation, and model application. The code and worksheet are divided into parallel ‘sections’ that guide students to run code and answer questions about the code, cardiac system, or results of the code concurrently. Students also enter results produced by their code into a google spreadsheet that allows for comparison, summary, and discussion of results across teams. The worksheet, code, and google spreadsheet are included in the supplemental materials and labeled COMPRESSED.

The activity begins by introducing students to a medical case - a 63 yr old woman with a complex medical history who is experiencing declining indicators of cardiac and respiratory function. In the worksheet, students are asked to annotate an provided ECG for markers of AFib. Each team then runs a section of code that loads data from a web archive, splits that data into test and training data, and plots three ECGs for teams to visually diagnose. Two of the signals (one normal, and one AFib) were common among all of the teams to encourage collective decision-making and checking of their answers. The third is drawn randomly and teams enter a diagnosis based on their interpretation of the ECG into the Google spreadsheet. The spreadsheet then compares each diagnosis to the expert diagnosis from the PhysioNet data set (Figure 1).



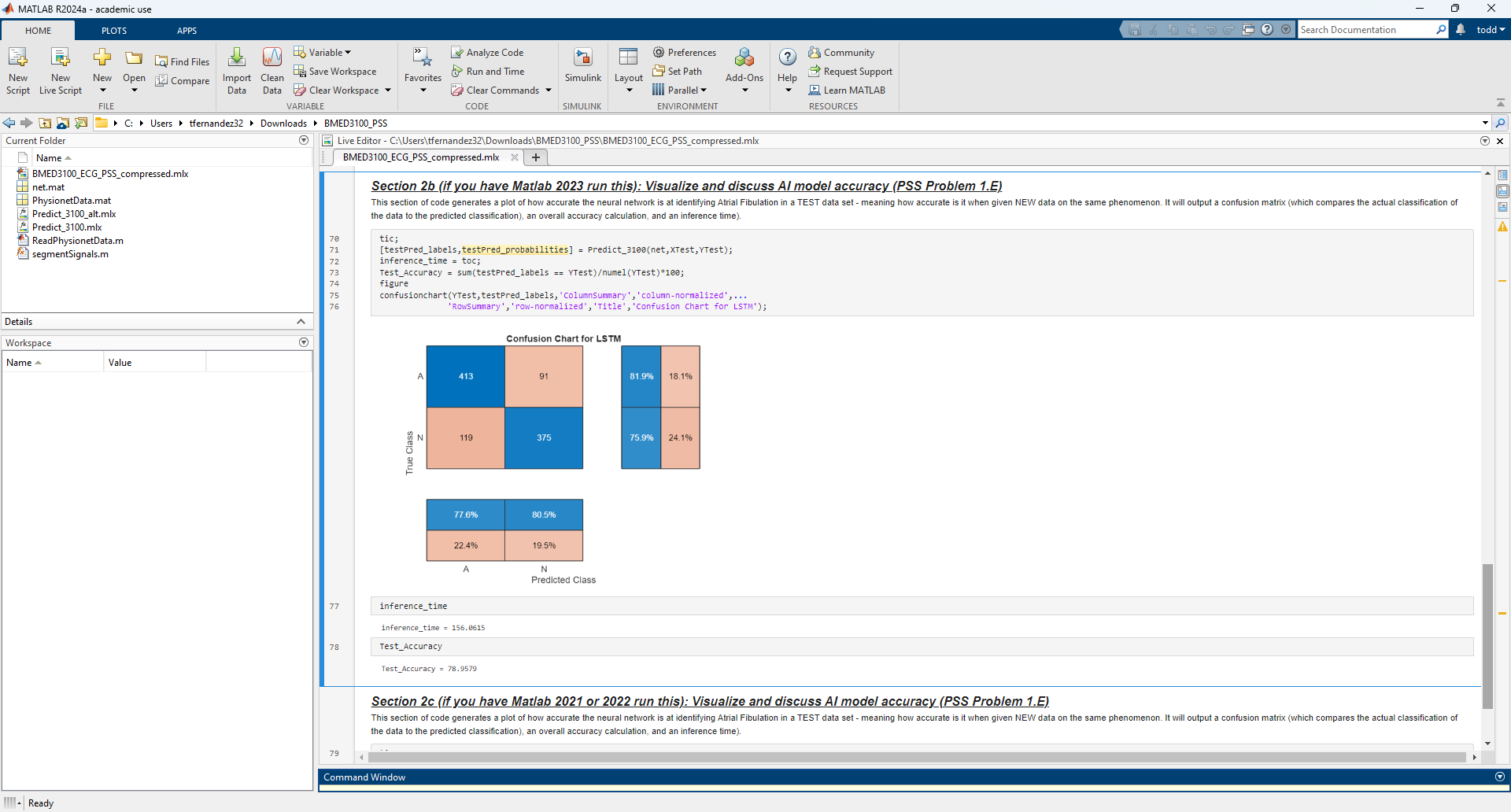
**Figure 1**. A view of the Google spreadsheet used by students running the compressed Version 2 of the activity. Students select their diagnosis for three, two of which are common among all teams (including ECG 4 and ECG 1), and one randomly-selected for each team by the code (ECG X). ECG 4 was labeled in the data set as AFib by experts, while ECG 1 was labeled normal.

The worksheet asks teams to discuss their diagnosis accuracy, the visual cues they used in their diagnosis, and to relate AFib back to the medical case. When most teams have completed this step, the diagnosis indicators of the ‘common’ ECGs are discussed as a course. Then, the instructor shows a summary of the diagnostic accuracy from the randomized ECGs, which is discussed and compared to literature on different types of clinicians (Figure 2). We found that students were surprised that their accuracy in diagnosing AFib (which, we note, was done as a team and also may have been impacted by class-wide peer learning, see Discussion) was more accurate than most groups of clinicians. As statistics is not a prerequisite for this class, the discussion here also offered a brief conceptual introduction to sensitivity and specificity and the implications (e.g., for clinical decision making) of both false positives and false negatives.

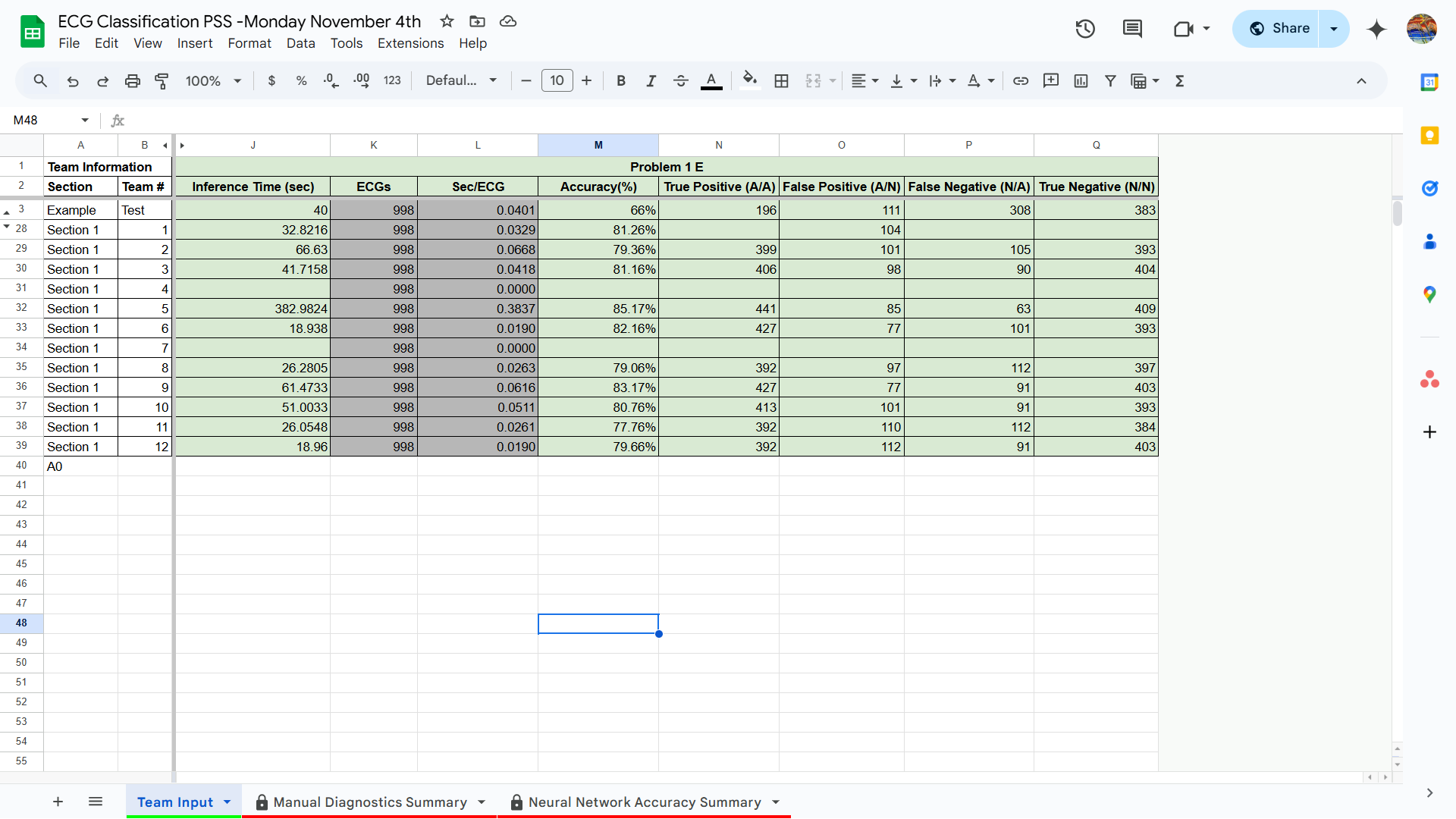


**Figure 2**. Summary of student accuracy from diagnosing randomized ECGs and comparison to accuracy amongst groups of clinicians.

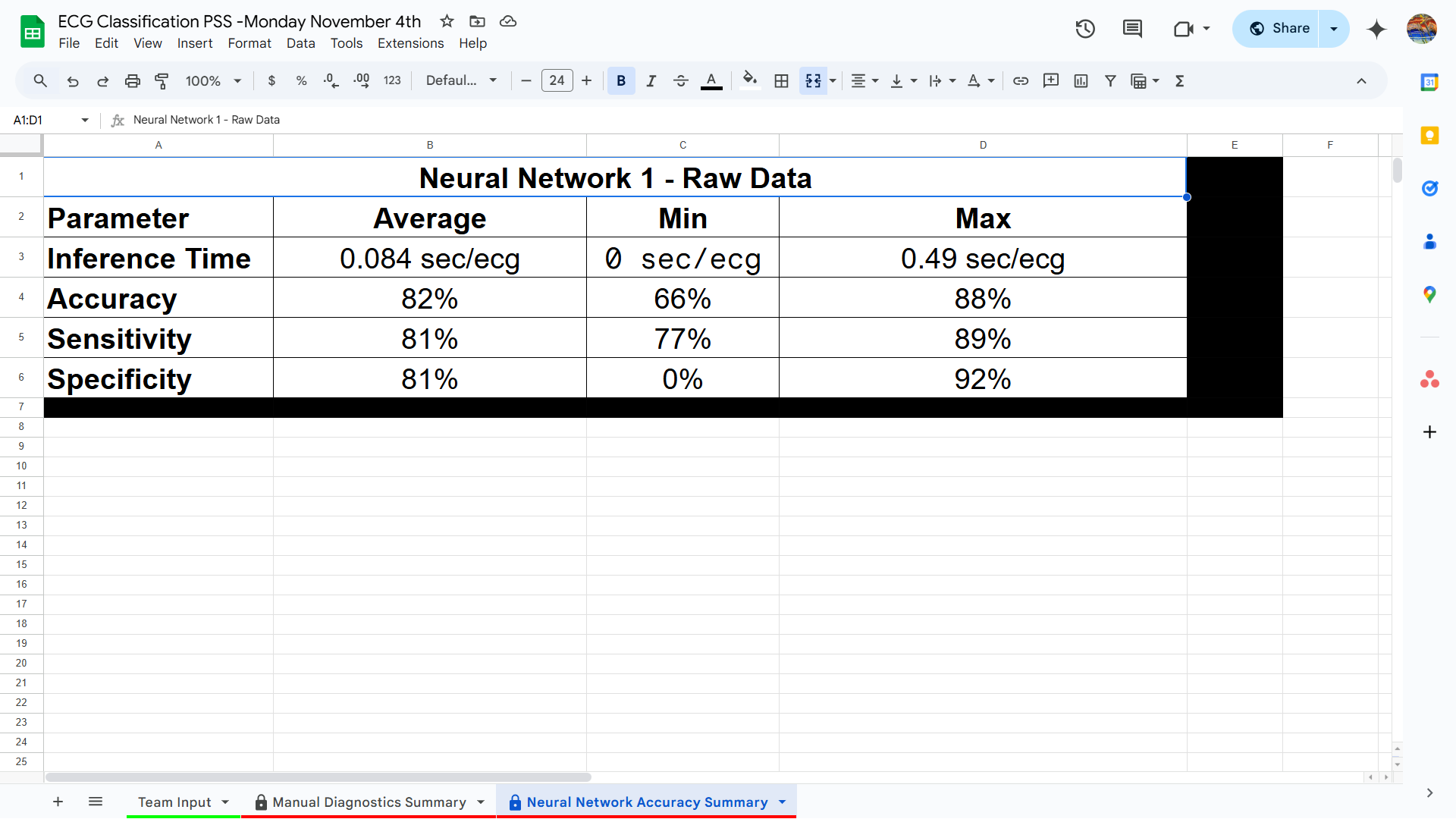
Students are then instructed to run section 2 of the code, which runs an inference process that classifies all of the raw ECG signals. This is run on a model pre-trained by the instructors on a GPU-based cluster resource, and depending on the student laptop takes between two and five minutes to complete. Because MATLAB has a rapidly evolving support for machine learning models, the code includes three versions of the same functional steps to align with different versions of MATLAB students have installed. While the inference code is running, students are asked to translate their visual diagnosis of ECG signals into mathematics and discuss challenges in making a mathematical function from an ECG. When the code is complete, it generates a confusion chart, inference time, and test accuracy (Figure 3). Students then enter that information into the Google spreadsheet (Figure 4) and a cross class summary is again generated (Figure 5). Finally, students are asked to put themselves in the shoes of clinicians and asked whether they would rely on this diagnostic tool to guide treatment, and what physiological consequences may result from a treatment choice based on an incorrect diagnosis. The worksheet then continues with related physiology content about AFib treatment and links between the cardiac and pulmonary systems.



**Figure 3**. The confusion chart, inference time, and test accuracy reported to students  
after the raw ECG data is classified by the model provided to them.



**Figure 4**. View of the spreadsheet where student teams enter information from their model’s outputs (from Figure 3).



**Figure 5**. Summary of the performance of a neural network to diagnose AFib from raw ECG data. This summary was generated after three class sections to show teams the similarity and variance that jointly exist in machine learning training results.

The choices and structure of this implementation are intended to provide a minimum impact introduction to the use of ML tools in a physiological space. Specifically, this version was developed to show instructors in the class and across the curriculum the type of minimal implementation that was possible for the introduction of data skills in any course. In doing so, this version of the activity demonstrated to the course instructors how ‘extra’ or ‘different’ content such as statistics related to ECG classification, could further student understanding of core course content (e.g., the cardiac system) without requiring the removal or adjustment of any content. The activity here retained all of the questions of the original activity (i.e., pre-intervention version). The results discussed below suggest that students’ understanding of the core course content actually improved compared to the pre-intervention version. If others were interested, it would be possible to pre-train all models described in the full implementation (Version 1) and run similar inference code locally to expand this activity in order to include the other machine learning models.

## Full implementation

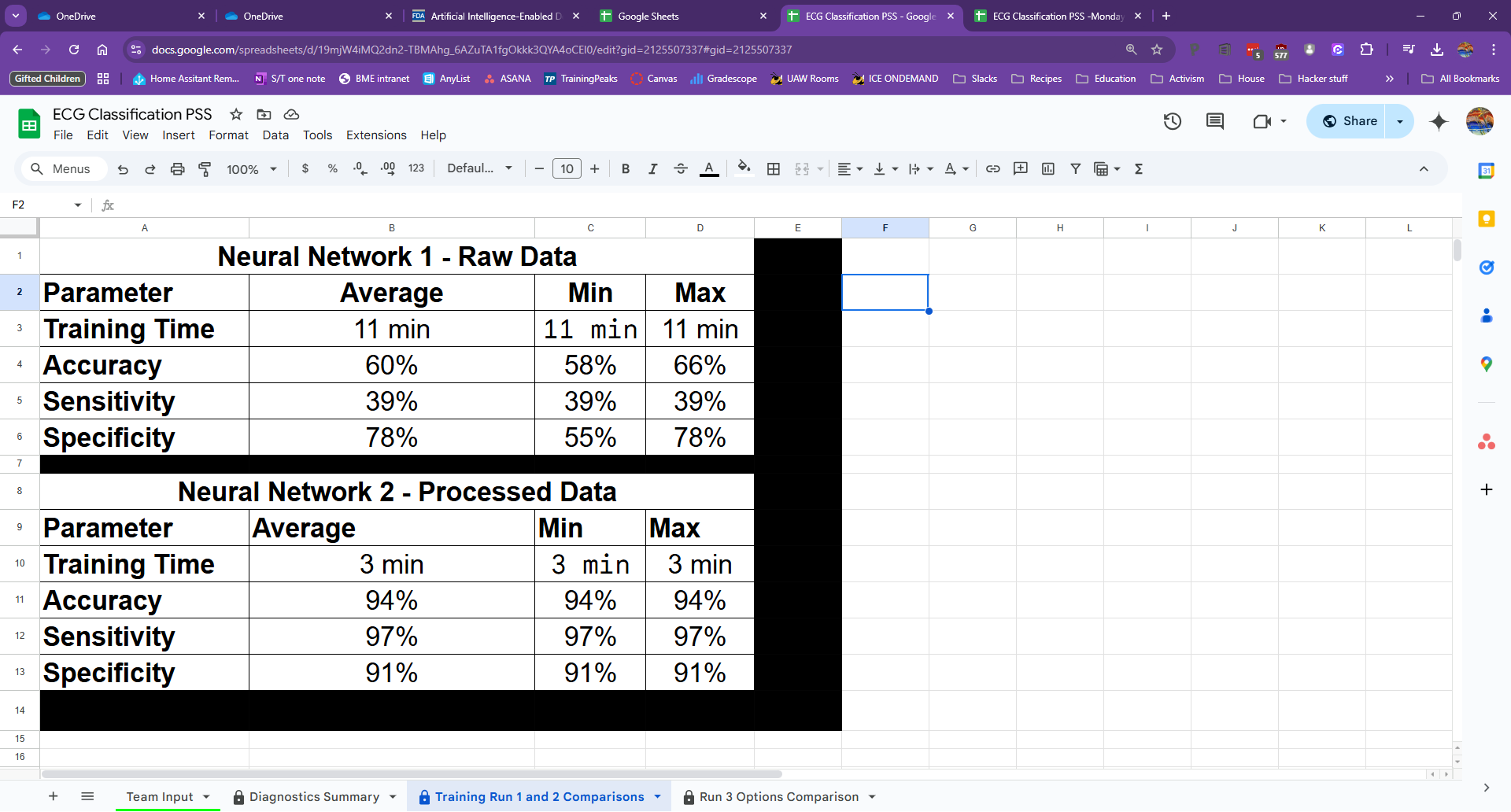
The full Version 1 of the implementation generally expands on the compressed interpretation in two ways: (1) students train models on a GPU-based compute cluster themselves as opposed to only performing inference locally and (2) students work to train and test three machine learning models on ECG. In our efforts at data-skills based curricular change, this version is meant to show the possibilities of changing a full 110 minute class period. By design, this implementation of the activity is meant to exchange deep work on details of cardiac physiology for a broad introduction to how cardiac physiology can be linked to other areas of engineering content - e.g., signals processing. It drops the context of a medical case and replaces it with framing in the development of a AFib detection product. This version of the activity was piloted with a group of 6 undergraduate and 2 graduate TAs from the course for feedback and evaluation. In the supplemental materials, all related files are labeled FULL.

The full implementation largely follows the compressed version of the activity to its completion and then continues. The primary difference where the two overlap is that in the full activity students take two extra steps prior to working with the models and their results. First, the students log into and load the data and code on a remote GPU based compute cluster (see Computational Resources and Collaboration section below for details). All code for the activity is then run on that cluster. Second, students open a highly simplified version (i.e., no comments, sections, one model) of the code on a single student laptop and run it. This simplified version provides a point of comparison of compute resources by running for the remainder of class and training only about 10% of one of the three models that students train on the cluster.

At the beginning of the activity, students perform the same random draw of ECGs, but are shown four random ECGs to diagnose as opposed to one random and two standardized ECGS. After entering their diagnoses into a similar spreadsheet, they are instructed to run the next section of the code, which begins training of the first ML model. Because training the model takes about 11 minutes on the NVIDIA H100 GPUs that we had available, we chose to use that time to have students complete other work in the activity. The class has a similar discussion of how they visually classified ECGs. Because the data set has a ratio of approximately 7:1 normal to AFib ECGS, the activity then walks students through a theoretical model where all signals are classified as normal - which would produce an accuracy of about 87%. This is used to discuss the need for balancing or replication in the data set and the potential for bias from data in ML models.

Once training and inference for the first model are complete, students enter results into their spreadsheet and answer questions similar to the compressed Version 2. In this version, students have the accuracy results for both training and test accuracy, with the model producing higher results for training (~90%) than test data (~55%). Combined with the manual calculation of the theoretical model (87%), the differences in accuracy enabled deeper discussion about machine learning models of ECG classification than the compressed version. In the reflection prompt and class discussion, students are asked to comment on the reasons and problems of a large difference in training and test accuracy as well as how to make sense of a model with lower accuracy than blind classifying all data as normal.

After analyzing the first model, students then repeat this process for a second ML model. Tracking with the base code from MATLAB, this model uses spectral analysis techniques to perform feature extraction on the ECG signals. It does so by calculating an overall power spectrum, instantaneous frequency, and spectral entropy of each signal. As with the statistics earlier, we note that deep knowledge of signals processing is not prerequisite knowledge for this course. The code first displays a graph with one AFib and one normal example of the processed signal (the signals are labeled this time). Students are asked to perform a visual examination similar to the raw ECGs and describe what differences, if any, they can see between AFib and normal for the processed signals. They then run the training and inference code that displays the same results as the first model, and they enter the same data into their spreadsheet. Training this model is significantly faster (3 min), and the training (~95%) and test accuracy (~95%) significantly higher than the raw model. The reflection prompts for this model differ by focusing on how feature extraction signals processing relate to the cardiac system and the cardiac behavior characteristic of AFib. The teams, and class, then focus on comparing the accuracy, efficacy, and practical challenges of using this model with doctors over one trained on raw ECGs using a summary of all teams’ results (Figure 6). The expanded time of this activity also enables a discussion of the sensitivity and specificity and the prioritization of those in product development.



**Figure 6**. View of the class-compiled data comparing the two trained networks from the full version of the activity.

The final portion of the full Version 1 activity involved students choosing from three options to improve the model that used raw ECGs. Each team could choose to either (1) make the network significantly (100x) bigger by adding another hidden layer and increasing the number of neurons in each layer, (2) resample the data to have 2 AFib signals for every normal signal during training, or (3) look at more signals in each training iteration. None of the options meaningfully improve the accuracy of the model, although options 1 and 2 significantly increase training time. The teams entered the results of the models in the Google spreadsheet similar to the other two training runs, which produces another summary similar to Figure 6. The activity wraps up with an overall reflection and a discussion of diagnostic usefulness.

## Computational resources and collaboration

We note that either version of this activity, but especially the full version, would not be possible without our on-campus access to significant computational resources. For those interested in a local version of the activity, we provide all of the pretrained models. However, we encourage others where possible to provide students the opportunity to perform the training themselves using institutional or commercial compute resources that they are likely to have never interacted with directly before. Below we describe the resources necessary and how we worked with our institutional compute resources team in planning and developing the activity.

[BLINDED] is an investment by [BLINDED] in AI/ML education, providing students with a set of supercomputing resources exclusively for student learning and exploration. The system [BLINDED] enables the expansion of AI/ML content in courses and will eventually be open for independent student investigations. [BLINDED - basically, it is free to students, part of our larger campus wide infrastructure to support training and use of supercomputing resources in both education and research]. To facilitate easy access to advanced computing by students with a wide variety of backgrounds, [BLINDED] uses Open OnDemand[18](https://paperpile.com/c/WJ14MC/CSfz), a graphical web portal for supercomputers. MATLAB and other applications are offered through simple user interfaces that allocate compute resources by interacting with the Slurm scheduler behind the scenes.

The overarching cluster runs on Red Hat Enterprise Linux 9 and is equipped with MATLAB version R2023b. That version includes the Parallel Computing Toolbox to facilitate GPU-based training and the Deep Learning Toolbox to leverage predefined bidirectional long short-term memory (LSTM) architecture support. However, it is different than the version of MATLAB (2024b) that the code we modified is based on and did not contain some functions in the Deep Learning Toolbox implemented in the MATLAB demo[17](https://paperpile.com/c/WJ14MC/6gL6). This forced two of the authors to rewrite the inference code using functions available in 2023b (see appendices). The use of 2023b relates to issues with memory leaks we have encountered when operating more recent versions of MATLAB on our architecture.

For the activity, [BLINDED] support staff allocated one node per team of four students scheduled for the duration of the in class session (or out of class testing). Students enrolled in the course were automatically provisioned with accounts on [BLINDED], including home directories and scratch storage [BLINDED]. To ensure availability during the lab exercise, a reservation was created for the duration of all lab sections with enough nodes for each lab group. Students were added to a POSIX group unique to the course, and the Slurm reservation used the magnetic feature to automatically place jobs submitted by these students during the reserved window onto the specified nodes, further simplifying the process for students.

Each node included 8 CPU cores, 256 GB of RAM, and one NVIDIA H100 SXM5 80 GB GPU. Under this configuration, a standard training of 30 epochs with a minibatch size of 200 on a preset bidirectional LSTM completes within approximately 10 minutes. The PhysioNet 2017 dataset used in the activity is less than 100 MB, and is provided as serialized, single-file train and test sets from MATLAB to the node. Given the datasets’ small size and format, we experienced no challenges regarding dataset storage or I/O performance across various storage media.

For readers without similar on campus resources, there are various commercial vendors who offer similar GPU computing services for MATLAB[19](https://paperpile.com/c/WJ14MC/LOpf) (Table 1). One good option is Amazon Web Services (AWS) with the ability to integrate with the MathWorks Cloud Center to provide a paralleled experience close to native. Additionally, MathWorks collaborates with Microsoft Azure Marketplace to offer preconfigured MATLAB instances through their software plans. Both AWS and Azure instances support GPU acceleration and can be configured with the Campus-Wide License for academic use. However, neither of these plans includes free trials or free-tier GPU resources. More generic cloud computing options are also available via deploying MATLAB’s Deep Learning Container[20](https://paperpile.com/c/WJ14MC/kmYV). However, this approach demands additional effort from course instructors to ensure proper configuration, conduct dry runs, and prepare additional instructions for students. We summarized the costs required for a single group to complete the two-hour session. It should be noted that miscellaneous costs, such as network bandwidth and data transfer, may apply but are not accounted for in the provided table. The extra time for the instructors to set and tune the materials beforehand is also not included.

**Table 1**. Commercial vendors and estimated costs for GPU computing services for MATLAB.

| **Commercial vendor option** | **Cost of one session per group ( 2 hours, US Dollar)** |
| --- | --- |
| AWS EC2 p5.48xlarge\*\* | ~$27 |
| Microsoft Azure ND96isr H100 v5\*\* | ~$25 |
| Estimated H100 remote rental\*\*\* | ~$4-10 |
| **Notes:** All instances feature H100 GPU and Red Hat Enterprise Linux (RHEL) operating system (Industry has higher rates with non-free Linux OS (i.e. RHEL). Actual costs likely to be slightly higher due to need for testing, setup, and overrun time.  \*\* Dedicated instances to meet BYOL (Bring Your Own License) requirement  \*\*\* Does not include MATLAB preconfiguration and would require the use of the docker hosting approach discussed above. | |

# Activity Evaluation

## Methods

As part of our curriculum-wide effort to improve student data skills, we are in the process of developing a survey-based approach to measuring data skills - especially students’ confidence in their skills, interest in careers using those skills, and perception of their relevance. We expect to detail development and testing of that survey in later work. The survey is generally based around the data skills described earlier in the paper. The survey has a large item pool and asks students about a set of 15 items (see appendix). Overall, 12 items were pre-selected and shown to all students with another 3 randomly selected from the item pool. Students are asked to respond to each general item in three forms: (1) Their confidence in their ability to apply each item, with guidance, to BME courses or work, (2) Their perception of the general applicability of the items to BME work, and (3) Their personal interest in jobs that make use of each of the items. We were particularly interested in the analysis of a subset of items most applicable to this activity and course context, specifically those about using preconfigured code and making sense of model credibility and applicability.

For this paper, our primary data source is from the compressed Version 2 of the activity. Students responded to the survey twice using a pre-post design. Students were sent a pre-survey recruitment email several days before the exercise. The pre-survey was closed the morning that the data skills exercise was scheduled. After the data skills exercise, students were sent a post-survey, and given a week to complete the post-survey. Students were offered a small amount of extra credit (1% of their final grade) by completing both the pre- and post-surveys, regardless of their permission for their surveys to be used in this study. Per IRB rules, they could also receive the extra credit by writing a reflection about the data skills exercise for the same credit. Overall, 142 students were invited to participate in the data collection. One student opted to write a reflection to receive the extra credit, due to missing the pre-survey deadline. 115 students completed both the pre- and post-surveys. This data collection procedure was approved by [BLINDED INSTITUTION’S IRB].

Our evaluation of the efficacy of the activity also includes comments from the course instructors and TAs as well as the participants in the full activity test run. Those are paired with observations from the developers of the activity in the results below. We chose not to collect any responses to the questions in the activity at this time, but plan to generally address student performance on the questions in future papers.

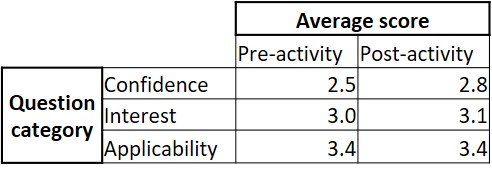
## Results

The survey results are separated into two parts. First, we address the results from the pre post survey of the compressed Version 2 of the activity. Second, we address student performance and qualitative observations from the full Version 1 of the activity.

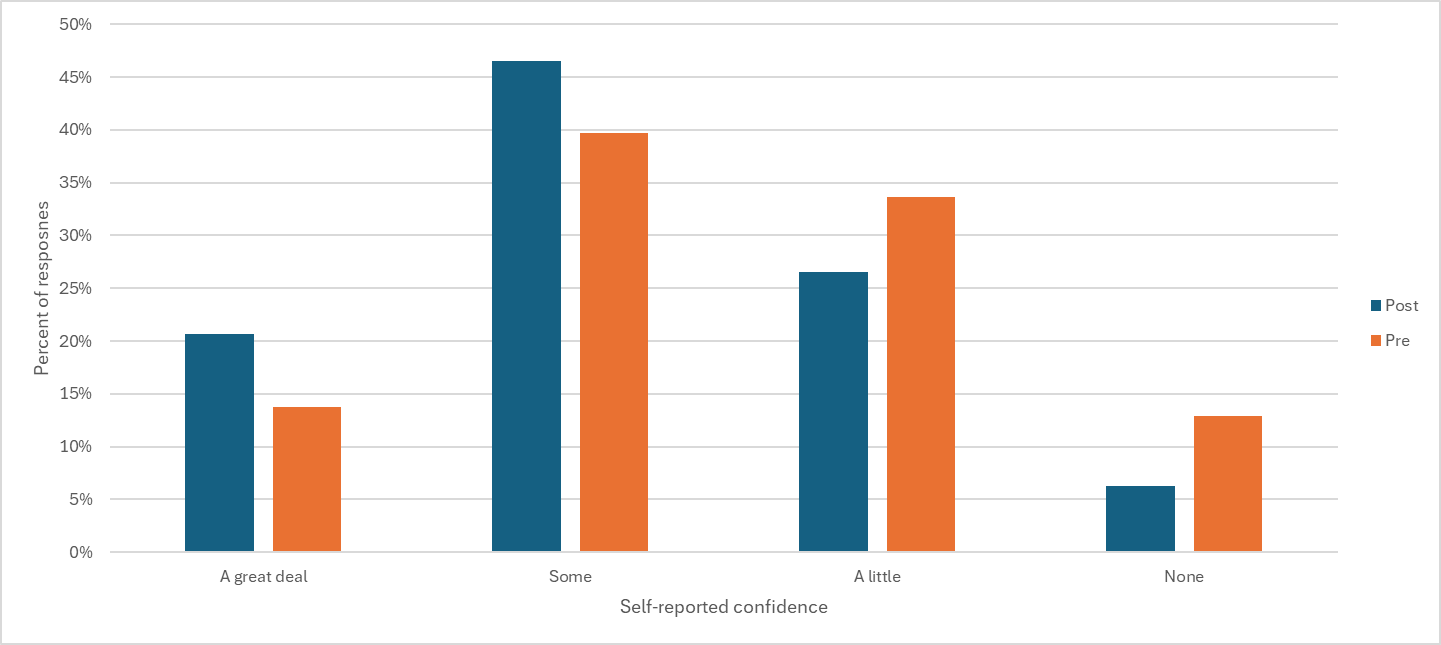
## Survey results from compressed activity

Overall, after accounting for those completing the survey but declining research consent, we received 133 pre responses and 128 post responses from the compressed activity. For this paper, we decided to primarily focus on aggregated responses in the three categories of survey question - confidence, applicability, and personal interest. This approach, in part, reflects a finding that we saw low question to question variation. We plan to further investigate this in future research. Here we focus primarily on descriptive statistics rather than psychometric analysis. The results showed small but meaningful shifts towards greater confidence, relevance, and interest (Table 2). Because this is a curricular wide project where impact is the result of multiple interventions, AND the survey is intended to measure that movement across a curriculum, we see this as normal and positive. However, the results did consistently move in the direction we expect - greater confidence and interest. Interestingly, we saw less change in the applicability component of the survey.

**Table 2**. Summary of scores from the three student survey question categories. Scores are on a 4 point scale,   
with a score of 4 corresponding to “*a great deal*” of confidence, interest, or applicability.



Overall, the survey results shifted towards greater confidence in their abilities related to the items asked, from the pre to the post (Figure 7). Notably, both the upper two categories (*a great deal, some*) increased their share of responses.The average response to the items about confidence increased about 0.3 points (from 2.5 to 2.8 - i.e., between *a little* and *some*) on our 4 point confidence scale. Of the 26 confidence items, 9 items had the median answer increase, the largest increase (average increased by 0.7pts, median went from *a little* to *some*) being in the item *“Find and use prewritten software code, in a software language that you are unfamiliar with, to perform data analysis tasks”*. In contrast, only 1 item (“Identify when it is appropriate to use different types of data analysis software e.g., a spreadsheet or programming language”) saw any decrease in either the mean or median response (-0.1 points, median from *some* to *a little*). While limited, we see this as evidence in support of our results’ credibility because of their literal connections to the activity. For this activity, students were shown and used prewritten software code from a source (MathWorks) with a vast library of similar activities - and their confidence in doing so went up. In parallel, they switched between three sets of analysis tools (their eyes, matlab, and google spreadsheet), but those choices were not highlighted by the activity in the interest of time - and scores went down.



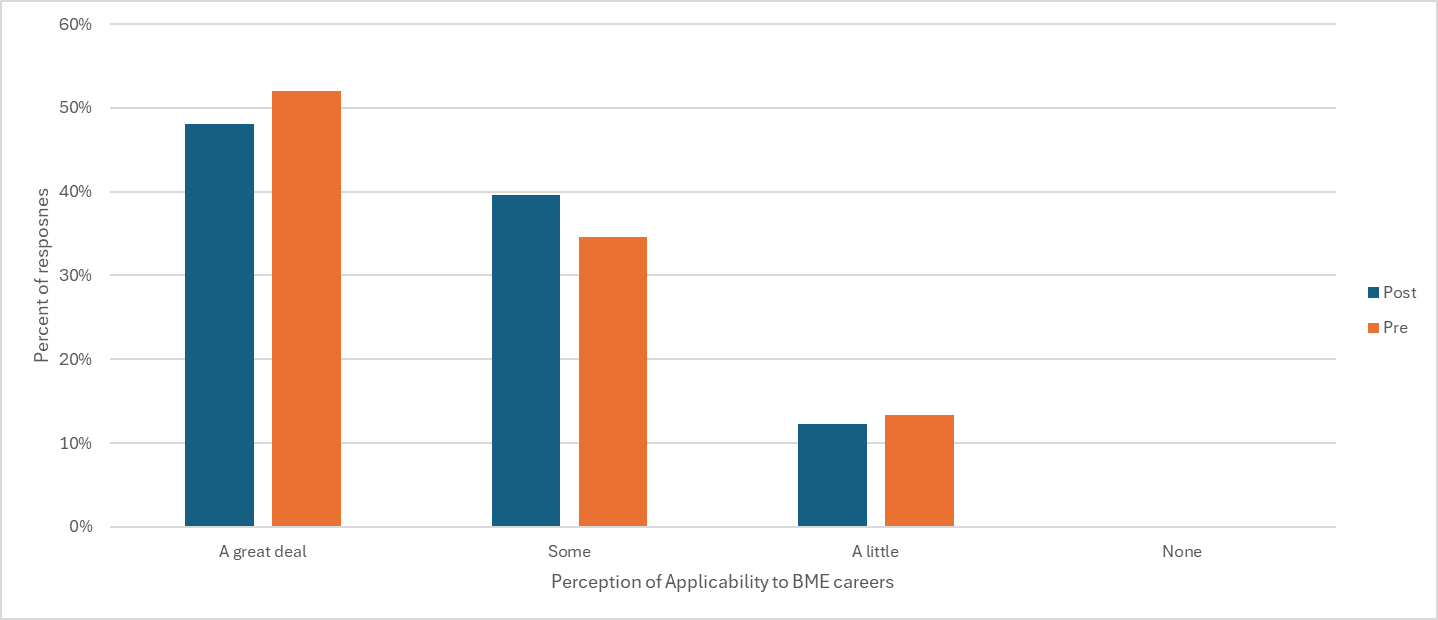
**Figure 7** Comparison of distribution of confidence between pre and post survey. Y axis is in percentage.

The survey also showed a shift, albeit smaller (3.0 to 3.1, +0.1 pts, with 3.0 meaning *some* interest), towards greater *interest in careers using these skills* from the pre to the post (Figure 8). Again, both the upper two categories increased their share of responses at the expense of *a little*. In both the pre and the post, across all items, there were zero *none* responsesto questions about interest or applicability. Of the interest items, 6 items had the median answer increase, with the largest change (average increased by 0.5pts, median went from *some* to *a great deal*) being in the item *“Justify decisions made in the process of applying statistical and machine learning techniques to data”*. The same number of items (6) had a decrease in the median answer. The largest decrease was the item “When evaluating procedures to collect data, identify populations that may be put at risk or excluded” whose average decreased by ~0.4pts (3.5 to 3.2, median went from *a great deal* to *some*). We do note that this item was one of the randomly presented (rather than ‘anchor’) questions so only received 42 total responses across the pre and post. As with the results from the confidence items, we see these as initial positive results for the activity (and the survey). The item with the largest growth in applicability, which is about justifying decisions, was a focus of both versions of the activity. Interestingly, the choice to focus on decisions and explanation as opposed to math or theory of neural networks was one of the most discussed aspects of developing the activity.

## 

**Figure 8** Comparison of distribution of interest between pre and post survey. Y axis is in percentage.

Shifts in the applicability ratings that students provided to the skills in the survey were also small - and significantly higher than the other two areas. The overall average of responses did not change between pre and post, and about midway (3.4) between *some* and *a great deal* of applicability to BME careers. None of the applicability items showed meaningful changes in the average response. Compared to the other two item groups, applicability saw a much smaller (and different) shift in the distribution (Figure 9). The *some* response category went up between pre and post, but both *a great deal* and *a little* went down. As a comparison, in the confidence items, the two lower groups each lost 7% of their responses pre to post, with the two higher confidence responses each gaining 7% - for a 14% shift upward. In contrast, there was a less than 5% total change among all the categories when it came to applicability. We consider all of these changes to be within measurement error at this stage and show no meaningful change in applicability responses. Our hypothesis is that these results show students’ attitudes about the relevance of AI and ML may be more highly formed than their knowledge of them - which would logically be more formative of their confidence and interest.



**Figure 9** Comparison of perceived applicability to BME careers between pre and post survey.

## Observations and comments from full activity

Because of the small number of volunteers who participated in the full activity (9 students, 4 teams), we did not collect survey data. Rather, we made qualitative observations about their actions and feedback. We noted several small technical and logistical details which might be helpful to other instructors. These included explaining to students the need to have two laptops per group, at least one of which had Matlab and the deep learning package installed. We anticipate in future versions having a clear single page of ‘preparation’ instructions to include that detail and minor details like the need to install a vpn to access our cluster. Further, the use of the run sections and inline output of the live editor was something we expected students to be familiar with because it is used in upper level BME courses. This manifested most often as looking at the command window not the live editor window for output - and being left unclear when computation was complete. An option here would be to enable Matlab’s built in training progress window. However, we had disabled for both versions as we found that, in combination with the VPN remote desktop, and some software versions, it caused errors. We assumed that meant they had been taught that in their Matlab based introductory coding class (taught by our CS department) but that was not true and more instructions there would have been useful. We also encountered a few Matlab file handling issues with Mac OSes that we plan to fix.

There was also a need to prep teaching assistants on some of the terms that were glossed over in the worksheet. For example, the first model trained is a Long term-Short Memory (LTSM) neural network. We had decided on that detail, an explanation of what an LTSM was, and the reasoning behind why an LTSM was not relevant to the core activity. However, the volunteers noted that if people had interest it would be important to have the TAs able to answer some questions in the moment instead of the only option being to go look at the extra resources listed to learn more. We plan to address this in the future by adding a ‘machine learning dictionary’ to the back of the assignment that covers topics like ‘what is a feature’ and ‘what is an epoch’ as a better bridge. This aspect also included opportunities that the volunteers identified to connect the content to that used in later courses (e.g., a later lab course studies heart rate variability, or connecting to our statistics course if this becomes a permanent activity.). However, a major summative note from the volunteers was the appreciation for something that linked the quantitative/math-heavy parts of our curriculum and the more information heavy-portions of our curriculum. While that does happen in later courses, which some of them had experienced, they appreciated the opportunity for it to happen earlier.

To the overall design, the volunteers noted that they perceived the experience of training models on the cluster to be of high value. They also appreciated having the comparison of cluster to local training efficiency. Conversely, the authors facilitating the activity noted a need for more precise, and more ‘higher level’ questions. We had kept the number of questions small to not overwhelm students, but found that they were often left waiting for training runs to complete - time that could be filled with more thinking and reflection. We expect to add questions about where this model could/could not be used in clinical support. We also observed that the options in the third run felt disconnected and random, and plan to restructure those to reflect three options within a single change - balanced, unbalanced, and over weighted - in the future.

# Discussion

Overall, we see this activity as generally achieving what we intended it to. That is, it made new learning about new material adjacent to physiology a part of the class. This both built confidence of students in their data skills abilities and better linked this course to other courses. As noted, we expect that changes in data skills will be small from a 30 minute exposure and are primarily interested in results across our curriculum as activities like this multiply. Overall, a critical but secondary implication for this activity was the impact it had on our faculty. Seeing what was changed grew the interest of a variety of faculty and courses in implementing similar activities. The low instructional overhead, the small course footprint, and the relative ‘coolness’ of getting to use the cluster all helped us build buy-in beyond the impact the activity had on students. None of that is to say that this activity was not without challenges or further work to do.

## Challenges and unexpected victories

There are two primary challenges that we experienced in implementing this activity: one practical and one technical. The practical challenge was faculty buy-in. Multiple classes in our curriculum are now in the process of developing interventions from the skills and principals discussed in this paper, with two courses ([this course] and [other course]) prioritized for 2024-2025. In that process we have generally found that the latter three pedagogical principals have been challenging for faculty. Specifically, the tradition of theoretically focused engineering teaching that is common in engineering is at odds with a willingness to focus on connecting information within a course, and an applications focus. For example, some instructors expressed concerns that students did not know the underlying mathematics of AI/ML tools. Others have expressed frustration at having to ‘fit’ new material into the courses. We note these because they situate our efforts at curricular change within fairly common objections within the literature. Through the development process we have reinforced the pedagogical principals to faculty as well as situated the work in any individual course in the larger ecosystem of AI/ML learning opportunities for students.

The technical challenge is related to the resources needed to replicate this activity. Because of course releases, the two authors who developed the activity were able to attend and assist all offerings of the compressed and full versions. That lowered any potential barriers to knowledge and training for faculty and TAs involved in the course - both real and perceived. We, as faculty at [R1 institution] are also aware that we have computational resources available to us for free that others may not. While the decision to pretrain models in the compressed version was initially meant to be a “backup”, meant to reduce the risk of the activity going wrong, we see it as useful to enable others without our resources to replicate the activity with lower cost (Table 1). If others are interested in replicating the activity, we include all of the pretrained models in our appendices [blinded for review].

We also saw unexpected ways in which learning was created. The primary one was how students used the google spreadsheet in the manual diagnostic portion of the activity. We included the google spreadsheet entry as a way to keep track of teams progress, but found them using it for self-directed learning. Once several teams had entered their diagnoses, it was very common for other teams to take other teams’ answers into consideration before entering their own diagnoses. They tended to answer the same way as other teams before them. When running the activity with a third group of students, we hid the answers of each team from the others. We noted that there was more variability in student diagnoses in this class. When using this activity, instructors can consider whether they do or do not want to show the work of each team to the whole class while the activity is in progress. We are inclined to make such work visible in the future, which can enable TAs and instructors to engage and ask questions about why teams changed answers to help uncover misconceptions or errors and reinforce learning about cardiac behavior. However, there is also an opportunity to force students to commit to a decision and then justify it. Our only definitive suggestion is to make clear that the diagnoses are not graded for accuracy.

## Future directions

As noted in the Results, we have a number of modifications we plan to make to the activity in future semesters. Practically, we plan to continue making the activity a part of the class. In future semesters we plan to take two primary actions related to research and assessment of it. The first is to collect students' answers to the activity questions to enable us to more directly evaluate learning. Second, we plan to work with the instructors to either offer the full version of the activity across the class or to have some sections offer each version to allow for a better comparison.

# Acknowledgment

We thank the instructors of the course [BLINDED], students, and TAs for their engagement and participation. We also acknowledge that the development and implementation of this activity was funded in part by a grant from the [BLINDED]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of [BLINDED].

# 

# References

1. [Committee on Envisioning the Data Science Discipline: The Undergraduate Perspective, Computer Science and Telecommunications Board, Board on Mathematical Sciences and Analytics, et al. *Data Science for Undergraduates*. National Academies Press; 2018.](http://paperpile.com/b/WJ14MC/wklx)

2. [Yavuz FG, Ward MD. Fostering undergraduate data science. *Am Stat*. 2020;74(1):8-16.](http://paperpile.com/b/WJ14MC/9JxR)

3. [Hicks SC, Irizarry RA. A guide to teaching data science. *Am Stat*. 2018;72(4):382-391.](http://paperpile.com/b/WJ14MC/kpHn)

4. [Sarp S, Kuzlu M, Popescu O, Jovanovic V, Acar Z. Development of a data science curriculum for an engineering technology program. In: *2023 ASEE Annual Conference & Exposition Proceedings*. ASEE Conferences; 2024. doi:](http://paperpile.com/b/WJ14MC/p4to)[10.18260/1-2--43109](http://dx.doi.org/10.18260/1-2--43109)

5. [Scott R, Frickenstein A, Wilhelm S. Longitudinal analysis of strategies for improving biomedical engineering student knowledge of career paths and desired skillsets. In: *2024 ASEE Annual Conference & Exposition Proceedings*. ASEE Conferences; 2024. doi:](http://paperpile.com/b/WJ14MC/1NhO)[10.18260/1-2--47751](http://dx.doi.org/10.18260/1-2--47751)

6. [Cassandra Sue Ellen Jamison Annie AnMeng Wang Aileen Huang-Saad Shanna R. Daly Lisa R. Lattuca". BME Career Exploration: Examining Students’ Career Perspectives. In: *2021 ASEE Virtual Annual Conference Content Access*. ASEE Conferences; 2021.](http://paperpile.com/b/WJ14MC/8Ket)

7. [Frow E, Caplan M. Promoting career reflection among freshman BME students. In: *2017 ASEE Annual Conference & Exposition Proceedings*. ASEE Conferences; 2018. doi:](http://paperpile.com/b/WJ14MC/LcUw)[10.18260/1-2--28771](http://dx.doi.org/10.18260/1-2--28771)

8. [Mikalef P, Giannakos MN, Pappas IO, Krogstie J. The human side of big data: Understanding the skills of the data scientist in education and industry. In: *2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE; 2018:503-512.](http://paperpile.com/b/WJ14MC/gUbY)

9. [Chi MTH, Wylie R. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educ Psychol*. 2014;49(4):219-243.](http://paperpile.com/b/WJ14MC/WVSt)

10. [Kember D, McKay J, Sinclair K, Wong FKY. A four‐category scheme for coding and assessing the level of reflection in written work. *Assess Eval High Educ*. 2008;33(4):369-379.](http://paperpile.com/b/WJ14MC/m6GR)

11. [Streveler RA, Brown S, Herman GL, Montfort D. Conceptual change and misconceptions in engineering education: Curriculum, measurement, and theory-focused approaches. In: *Cambridge Handbook of Engineering Education Research*. Cambridge University Press; 2015:83-102.](http://paperpile.com/b/WJ14MC/mbMb)

12. [*A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives*.](http://paperpile.com/b/WJ14MC/Fyi9)

13. [Ball KL. Foundations in physiology: an introductory course using the core concepts. *Adv Physiol Educ*. 2023;47(3):501-507.](http://paperpile.com/b/WJ14MC/GImv)

14. [Semsar K, Brownell S, Couch BA, et al. Phys-MAPS: a programmatic physiology assessment for introductory and advanced undergraduates. *Adv Physiol Educ*. 2019;43(1):15-27.](http://paperpile.com/b/WJ14MC/EalG)

15. [Doux J, Waller A. The problem solving studio: An apprenticeship environment for aspiring engineers. *Advances in engineering education*. 2016;5.](http://paperpile.com/b/WJ14MC/iJsl) <https://eric.ed.gov/?id=EJ1122000>

16. [Clifford GD, Liu C, Moody B, et al. AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017. Published online February 1, 2017. doi:](http://paperpile.com/b/WJ14MC/e1MI)[10.13026/D3HM-SF11](http://dx.doi.org/10.13026/D3HM-SF11)

17. [Classify ECG Signals Using Long Short-Term Memory Networks. Accessed December 9, 2024.](http://paperpile.com/b/WJ14MC/6gL6) <https://www.mathworks.com/help/signal/ug/classify-ecg-signals-using-long-short-term-memory-networks.html>

18. [Hudak D, Johnson D, Chalker A, et al. Open OnDemand: A web-based client portal for HPC centers. *J Open Source Softw*. 2018;3(25):622.](http://paperpile.com/b/WJ14MC/CSfz)

19. [Run MATLAB using GPUs in the Cloud. Accessed January 13, 2025.](http://paperpile.com/b/WJ14MC/LOpf) <https://www.mathworks.com/help/parallel-computing/run-matlab-using-cloud-gpus.html>

20. [MATLAB Deep Learning Container on Docker Hub. Accessed January 13, 2025.](http://paperpile.com/b/WJ14MC/kmYV) <https://www.mathworks.com/help/cloudcenter/ug/matlab-deep-learning-container-on-docker-hub.html>

# Supplemental materials

## Full Version 1 Activity (partial and blinded)

**[BLINDED]**

**Group Number: \_\_\_\_\_\_\_**

**Names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objectives:**

1. Use knowledge of ECG/EKG to identify events in Cardiac cycle.
2. Understand how clinicians perform disease diagnosis via ECG.
3. Evaluate accuracy of Machine learning models to diagnosis cardiac events via ECG
4. Understand how signals processing and concepts from calculus and mathematics modeling can improve the accuracy of algorithms for diagnosing ECG signals.
5. Understand pros and cons of large data and visual approaches to cardiac disease diagnosis

**The following questions are based an open data challenge and example code from MathWorks**

**Data source:** PhysioNet/Computing in Cardiology Challenge 2017   
*Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.*   
https://physionet.org/content/challenge-2017/1.0.0/

**Code Basis:** Classify ECG Signals Using Long Short-Term Memory Networks   
*Matlab Example Exercise* (<https://www.mathworks.com/help/signal/ug/classify-ecg-signals-using-long-short-term-memory-networks.html#ClassifyECGSignalsUsingLSTMNetworksExample-1>)

**Data Explanation:**

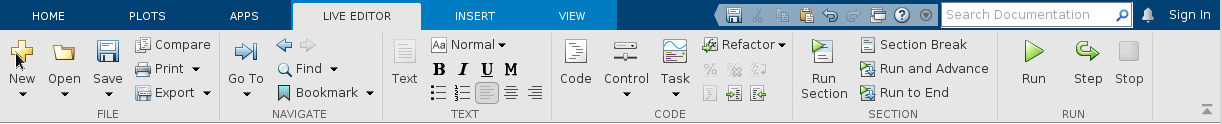
The data from the PhysioNet Challenge is single lead ElectroCardioGrams (ECGs). That means, a single electrical signal is recorded from each patient. The data consists of a set of ECG signals sampled at 300 Hz and are between 30 and 60 seconds in length. While the data contains many potential signals that can generate insights about ECG quality or cardiac behavior, each ECG was labeled by experts using on of four different classes: Normal (N), AFib (A), Other Rhythm (O), and Noisy Recording (~). The Matlab code simplifies the data by removing anything besides AFib (A) and Normal (N) signals for this exercise. The data set you will use contains 5788 of the 8528 ECGs in the larger challenge.

Atrial Fibrillation (AF) is a “tachyarrhythmia characterized by predominantly uncoordinated atrial activation with consequent deterioration of atrial mechanical function”. AF is the most common cardiac arrhythmia, Despite the enormity of this problem, automated detection of AF remains problematic. Algorithms to detect AF generally take one of two approaches: atrial activity analysis-based or ventricular response analysis-based methods. Atrial activity analysis-based AF detectors analyze the absence of P waves or the presence of fibrillatory *f waves*. Atrial activity analysis-based AF detectors can achieve high accuracy if the recorded ECG signals are high quality, but are easily degraded by noisy data. In contrast, ventricular response analysis is based on the predictability of the inter-beat timing of the QRS complexes in the ECG. RR intervals are derived from the most obvious large amplitude feature in the ECG, the R-peak, the detection of which can be far more noise resistant. It is worth noting that AF detectors that combine both atrial activity and ventricular response could provide an enhanced performance by combining independent data from each part of the cardiac cycle.

**Setup instructions to run local and cluster code**[BLINDED]

Part 1: Loading data and looking at ECGs yourself

Read the introduction to the code (briefly)…it includes the note to use the run section button, NOT the run button to run code. The code is divided into numbered sections (*Section #*). Shape



**Run Section 1 of the code** – this will load the raw ECG data into Matlab. You should end up with 5 total variables in your workspace, as shown below:   
A screenshot of a computer

Description automatically generated

**If your workspace DOES NOT show this after about 15 seconds** – please ensure you did step 5 on the previous page   
If you are still stuck, put your hand up for help.

**Section 1 of the code outputs a summary of the data set telling you how many signals labeled as AFib and NOT AFib there are in the data. Write those numbers down. We’ll use it later**

**Run Section 2 of the code -**this will output four (4) random ECGs as plots. As a team, inspect them and classify each one as either AFib or NOT AFib. Enter the signal number *(from the plot title, ECG#\_\_\_\_)* and your diagnosis of each ECG into appropriate row (section and team number) in the google spreadsheet.

**When you are done, as a team discuss the questions below.** As everyone finishes, you will discuss them as a class:

* What in the ECG signals did you look for to make your diagnosis?
* What made diagnosis challenging?
* How confident are you in your diagnoses?

Part 2: Training a neural network on the ECG Signals

In this part you will configure and train a relatively simple neural network on the raw ECG signal data. That involves running three sections (numbers 3, 4, and 5) of the code.

**Run Section 3 of the code** – This section of code will address the 7:1 ratio of not AFib to AFib data and prepare it for use. It does so by replicating the AFib signals until there is an 1:1 ratio and randomly dividing them into three groups (Train, Test, Validate). When the code is done it will print the number of signals in each group. Look at those numbers – what do notice about test vs. training?

**Run Section 4 of the code** – this section defines and then starts to train the first neural network. It should take between 10 and 15 minutes. When it is done it will print a summary of your newly trained neural network in the code window. While its running answer the three questions below:

1. Earlier you recorded the number of Not AFib and AFib signals in the original data set. Using those numbers, we want you to calculate a hypothetical accuracy if we just decided to be bad clinicians and classify all the ECGs as Not AFib. We could also do the opposite and classify every ECG as positive – calculate that hypothetical accuracy as well.   
   ***Formula:*** *Accuracy = # correctly classified signals / # of total signals*
2. How do those accuracies relate to the code in section 3 where we fixed the 7:1 ratio?
3. In the graphic that pops up, called the training progress monitor, the validation accuracy (basically an internal test of the network’s accuracy) and the training accuracy reflect the model’s accuracy as it learns. What do you notice about those lines?

**Report basic details of the model** - When the model is done training done it will output some basic data. Please note the number of parameters and not how many iterations the neural network went through in training. Record those in the spreadsheet.

**Run Section 5:** Section 5 uses the network to test the accuracy of your network. The results is an overall accuracy for this training network and a ‘confusion chart’. This takes some time as it passes each ECG through the trained neural network. When done, record the accuracy and raw values (not percentages, see green box at right) from the confusion chart into the google spreadsheet. 

A diagram of a number of ecgs

Description automatically generated

**When you are done, as a team discuss the questions below.** As everyone finishes, you will discuss them as a class:

* What is the problem with having high test accuracy and low training accuracy?
* Why might raw ECGs signal be challenging to fit a mathematical signal to?
* What do you think would make this model better?

## [activity continues from here]

## Compressed Version 2 Activity Code (adjusted to blind)

# Classifying ECGs using neural networks on your laptop

This code will use a pretrained neural network to classify ECGs using your personal laptop. The code is divided into two sections that you will run to answer the questions in the PSS. This is based on a using electrocardiogram (ECG) data from the PhysioNet 2017 Challenge. You can find that larger example here: [Classify ECG Signals Using Long Short-Term Memory Networks with GPU Acceleration](https://www.mathworks.com/help/signal/ug/classify-ecg-signals-using-long-short-term-memory-networks-with-gpu.html). [BLINDED]

# Major Note: You should run this code in SECTIONS

That means go to the **live editor toolbar** then you need to use **RUN SECTION not RUN.** . Each code section will output information that you need to answer the questions on the PSS worksheet.

## Section 1: Load data, configure data, display ECGs (Problem 1.A and 1.B)

*This section loads all of the files you need and will display several ECGs that you will use to do manual diagnostics*

### **Load Data**

this section loads all the variables you will need and does basic configuration

clear all

clc

%load everything we need

load PhysionetData

load net

[Signals,Labels] = segmentSignals(Signals,Labels);

afibX = Signals(Labels=='A');

afibY = Labels(Labels=='A');

normalX = Signals(Labels=='N');

normalY = Labels(Labels=='N');

The code below summarizes the 'labels' - meaning it tells you how many AFib and how many normal signals there are. The ratio is about 7:1 Not AFib to AFib, which means there are far fewer AFib signals. This will come into play in a minute.

summary(Labels)

**A** 718

**N** 4937

### **Setup data**

This section divides the data into equal test and training data sets

rng("shuffle")

[trainIndA,testIndA] = dividerand(length(afibX),0.8,0.1,0.1);

[trainIndN,testIndN] = dividerand(length(normalX),0.8,0.1,0.1);

XTrainA = afibX(trainIndA);

YTrainA = afibY(trainIndA);

XTrainN = normalX(trainIndN);

YTrainN = normalY(trainIndN);

XTestA = afibX(testIndA);

YTestA = afibY(testIndA);

XTestN = normalX(testIndN);

YTestN = normalY(testIndN);

The dataset is imbalanced. To achieve a similar number of AFib and Normal signals, repeat the AFib signals seven times. This is important to prevent bias. If we did not do this, the model could achieve 87% accuracy just by blindly classifying every signal as ***not AFib***. This is a common problem in diagnostic testing...if the thing you are testing for is rare in your population, high ***accuracy*** is driven primarily by the number of negative results given to negative tests no matter what happens with the other tests.

XTrain = [repmat(XTrainA,7,1); XTrainN];

YTrain = [repmat(YTrainA,7,1); YTrainN];

XTest = [repmat(XTestA,7,1); XTestN];

YTest = [repmat(YTestA,7,1); YTestN];

### **Print ECGs for diagnostics**

This code generates three plots of ECG signals. The first two plots are the same for everyone, the last is random.

rand\_1 = 4;

rand\_2 = 1;

rand\_3 = randi(length(Labels));

ecg\_1 = Signals{4};

ecg\_2 = Signals{1};

ecg\_3 = Signals{rand\_3};

subplot(3,1,1)

plot(-ecg\_1)

title(sprintf('ECG #%d',rand\_1))

xlim([4000,5200])

ylabel('Amplitude (mV)')

subplot(3,1,2)

plot(ecg\_2)

title(sprintf('ECG #%d',rand\_2))

xlim([4000,5200])

xlabel('Samples')

ylabel('Amplitude (mV)')

subplot(3,1,3)

plot(ecg\_3)

title(sprintf('ECG #%d',rand\_3))

xlim([4000,5200])

xlabel('Samples')

ylabel('Amplitude (mV)')

## Section 2a (if you have Matlab 2024 run this): Visualize and discuss AI model accuracy (Problem 1.E)

This section of code generates a plot of how accurate the neural network is at identifying Atrial Fibrillation in a TEST data set - meaning how accurate is it when given NEW data on the same phenomenon. It will output a confusion matrix (which compares the actual classification of the data to the predicted classification), an overall accuracy calculation, and an inference time).

classNames = categories(YTest);

scores = minibatchpredict(net,XTest,InputDataFormats="CTB");

testPred = scores2label(scores,classNames);

inference\_time = toc;

Test\_Accuracy = sum(testPred == YTest)/numel(YTest)\*100;

figure

confusionchart(YTest,testPred,'ColumnSummary','column-normalized',...

'RowSummary','row-normalized','Title','Confusion Chart for LSTM');

inference\_time

Test\_Accuracy

## Section 2b (if you have Matlab 2023 run this): Visualize and discuss AI model accuracy (Problem 1.E)

This section of code generates a plot of how accurate the neural network is at identifying Atrial Fibrillation in a TEST data set - meaning how accurate is it when given NEW data on the same phenomenon. It will output a confusion matrix (which compares the actual classification of the data to the predicted classification), an overall accuracy calculation, and an inference time).

tic;

[testPred\_labels,testPred\_probabilities] = Predict\_3100(net,XTest,YTest);

inference\_time = toc;

Test\_Accuracy = sum(testPred\_labels == YTest)/numel(YTest)\*100;

figure

confusionchart(YTest,testPred\_labels,'ColumnSummary','column-normalized',...

'RowSummary','row-normalized','Title','Confusion Chart for LSTM');

inference\_time

inference\_time = 156.0615

Test\_Accuracy

Test\_Accuracy = 78.9579