Paper 1(Algorithms for EEG Datasets):

- Systematic literature review focused on the use of machine learning in EEG
- Aims to empower CS students interested in BCI with knowledge and insights about the field
- CNN, RNN, Transformer, SVM, KNN, RF are the recommended machine learning algorithms for CS students to begin their exploration of EEG analysis
- Motor Imagery, Seizure Detection, and Emotion Detection are the top three EEG tasks in BCI research
- For Motor Imagery, the most commonly used algorithms are CNN, RNN, Transformer, SVM, RF.
- For Emotion Detection, the most commonly used algorithms are CNN, RNN, SVM, KNN.
- For Seizure Detection, the most commonly used algorithms are CNN, RNN, Transformer, KNN.
- DEAP, EEGEyeNet, CHB-MIT, SEED, and BCI Competition IV are the key EEG datasets for CS students starting in EEG analysis.
- DEAP for human affective state analysis, BCI Competition IV for Motor Imagery, and CHM-MIT for seizure analysis.
- Step-by-step guide for getting started with BCI research

Paper 2 (Predicting Pushback of Flights):
Significant Points:

- State the point of the research paper which is predicting the time of push of flights, including defining 'pushing flights', The need to optimize airport capacity and runway usage
- Predicting pushback time is important for air traffic management
- A gradient-boosting decision tree model is used, which is trained on data on weather, airport activity, airline, and aircraft characteristics.
- Discusses process used in the machine learning pipeline, including data preprocessing, feature extraction, model training, and validation as part of improving model performance and interpretability
- Features of the dataset used for training (Airport, time, aircraft, weather)
- Talk about the training process of the predictive model, whether global or local models were used, and how they were analyzed like using mean absolute error (MAE)
- Discusses difference in training with data from individual airports (local model) vs all airports (global model)
- Mean absolute error for evaluating model's estimate of pushback time
- Compares the performance of their model against a baseline, reducing baseline of 14.3 minutes to 10.7 minutes (MAE of pushback time based on other factors)

Paper 3 (Analyzing origin of biases in facial recognition analysis)

- It is critical that facial expression recognition models perform consistently and equitably across diverse populations due to their widespread usage; At present, this is not reality; “existing public FER models demonstrate biases in the faces of diverse populations”
- This paper analyzes these biases; generated an artificial facial expression dataset (940 faces) which researchers could manipulate to “isolate the impact of distinct manipulations on our model” and “better understand” the biases
- The goal is to investigate skin color biases, their complexity and non-linearity and study the impact of skin color distribution in the training set to highlight the necessity of dataset diversity and distribution
- The results reflected the existence of a skin color bias within FER models, displaying consistent disparities between skin tones in this testing environment
- Limited due to this being a small dataset lacking in diversity (ethnicity, morphology, pose, lighting, etc.); all faces were European and had the same morphology
- Also only evaluated a single model and a few variables–the results only apply to that model

Paper 4(The Friendship Paradox: An Analysis on Signed Social Networks with Positive and Negative Links):

- Significant Points:
    - Network Topology: The relationships between the individuals involved in the system (Social Platform)
    - Friendship Paradox: On average, an individual’s friends have more friends than that individual. A node is more likely to be a neighbor of a node with many neighbors (i.e., high degree), compared to being linked to a node with only a few edges (i.e., low degree).
    - First Order: $\psi_{i,1}^{\alpha\beta} = \frac{1}{N_{\alpha}(v_i)} \sum_{v_j \in N_{\alpha}(v_i)} 1\,(|N_{\beta}(v_j)| > |N_{\beta}(v_i)|)$
    - Second Order:
      $$\psi_2^{\delta} = \frac{1}{N} \sum_{i=1}^{N} 1\left(\left(\frac{1}{|N_{+}(v_i)|} \sum_{v_j \in N_{+}(v_i)} |N_{\delta}(v_j)|\right) > \left(\frac{1}{|N_{-}(v_i)|} \sum_{v_k \in N_{-}(v_i)} |N_{\delta}(v_k)|\right)\right)$$
    - Datasets: Bitcoin Alpha, Wiki Elections, Honduras Village, Slashdot, Epinions
    - The set of users that view a node positively and the set of users the node views positively are likely to have on average even more incoming and outgoing positive reviews.
    - Most nodes have both less friends and less enemies than its friends and enemies do.
    - Users who view a node negatively, are on average, likely to leave more negative reviews than the node does
    - Positive Trend: For most nodes, their friends have more friends than their enemies do.
    - Negative Trend: For most nodes, their friends also have more enemies than their enemies do.
    - Negative Trend is slightly weaker than the positive trend.

Paper 5(Autograding for Block-based Code):

- Designs a Computational Thinking course designed around Snap!, an introductory block-based programming language, to teach computer science fundamentals.
- Uses data from this course, which focuses on abstraction, algorithms, data representation, and documentation, to train and evaluate an automated evaluator (autograder) for Snap projects.
- Autograder measures repetition and encapsulation, failed to measure features such as data representation and documentation well,
- Compared to human graders, the autograder lacked the ability to highlight student improvement, create personalized alternative coding solutions, appreciate the artistic output of multimedia programs.
- Contribution: reports which features can be extracted from XML representation of a Snap! program, and how to analyze them. This can be generalized to other block-based programming languages.