GR5205 Final Project

Chen Liu(cl4682), Yewen Li(yl5888)

May 16, 2025

1 Abstract

This study examines the predictive power of clinical assessments versus neuroimaging features for classifying attention-deficit/hyperactivity disorder (ADHD) diagnosis and biological sex. Using the Women in Data Science Datathon 2025 dataset, which includes socio-demographic and behavioral questionnaire data alongside resting-state fMRI functional connectivity measures for over 1,500 children and adolescents, we trained several machine learning models (logistic regression, random forests, multi-layer perceptron, XGBoost, and LightGBM) to predict ADHD status and sex. Results reveal a significant contrast between the two classification tasks. ADHD diagnosis could be predicted with high accuracy using only clinical metadata, with questionnaire-based features dominating the feature importance. In contrast, predicting sex proved significantly more difficult: all models performed near chance, often defaulting to the majority class. These findings suggest that clinical assessments capture the core signals relevant to ADHD, whereas the neural connectivity patterns did not enhance diagnostic prediction. Moreover, the poor sex classification performance indicates that sex-related differences in brain connectivity are subtle and not easily captured by the available features.

The study underlines the importance of feature selection and modality in neurodevelopmental disorder prediction, and it highlights the need for more sensitive approaches to uncover sex-based brain differences.

2 Problem description

Attention Deficit Hyperactivity Disorder (ADHD) remains one of the most frequently diagnosed neurodevelopmental disorders, impacting individuals throughout childhood into adulthood. Traditional diagnostic criteria have predominantly been based on symptom presentations observed in male populations, leading to a potential oversight of distinctive manifestations of ADHD among females. Recent literature emphasizes a growing acknowledgment that ADHD symptoms are often subtler and less overtly hyperactive in female, potentially contributing to delayed or missed diagnoses.

Given the critical implications of these diagnostic disparities, it is imperative to enhance our understanding of how ADHD differentially manifests across sexes. This research leverages data from the WiDS Datathon 2025, facilitated by the Healthy Brain Network (HBN), a comprehensive initiative aiming to advance understanding of brain development and associated neurodevelopmental disorders. Utilizing machine learning and predictive modeling techniques, this project seeks to reveal diagnostic patterns specific to each sex.

3 Data Description

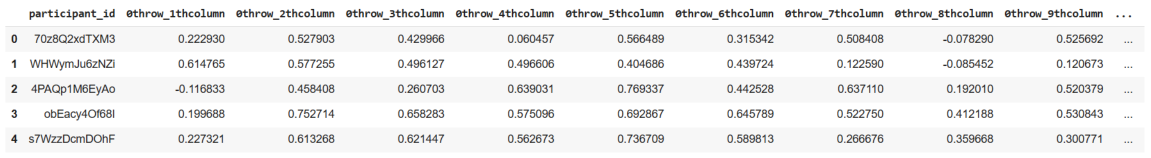
This study's datasets are provided by the Healthy Brain Network (HBN), a large-scale initiative that collects brain data to help understand neurodevelopmental disorders such as ADHD, to foster comprehensive research on neurodevelopmental conditions.

Our analysis incorporates three data sets:

3.1 Brain Functional Connectivity Data

This data set includes functional magnetic resonance (fMRI) connectome data, representing functional connectivity patterns between different regions of the brain. Each observation corresponds to a pairwise connection between brain regions, capturing the strength of their functional relationship.

This data set comprises more than 24.1 million observations.



	participant_id	0throw_1thcolumn	0throw_2thcolumn	0throw_3thcolumn	0throw_4thcolumn	0throw_5thcolumn	0throw_6thcolumn	0throw_7thcolumn	0throw_8thcolumn	0throw_9thcolumn	...
0	70z8Q2xdTXM3	0.222930	0.527903	0.429966	0.060457	0.566489	0.315342	0.508408	-0.078290	0.525692	...
1	WHWymJu6zNZi	0.614765	0.577255	0.496127	0.496606	0.404686	0.439724	0.122590	-0.085452	0.120673	...
2	4PAQp1M6EyAo	-0.116833	0.458408	0.260703	0.639031	0.769337	0.442528	0.637110	0.192010	0.520379	...
3	obEacy4Of68I	0.199688	0.752714	0.658283	0.575096	0.692867	0.645789	0.522750	0.412188	0.530843	...
4	s7WzzDcmDOhF	0.227321	0.613268	0.621447	0.562673	0.736709	0.589813	0.266676	0.359668	0.300771	...

Figure 1: fMRI connectome data

3.2 Socio-demographic Data

This dataset includes 12,130 observations, comprising variables such as age, gender, and parental education level of the participants. The variables include the following:

- *Basic_Demos_Enroll_Year* - Year of enrollment
- *Basic_Demos_Study_Site* - Site of phenotypic testing
- *PreInt_Demos_Fam_Child_Ethnicity* - Ethnicity of child
- *PreInt_Demos_Fam_Child_Race* - Race of child
- *MRI_Track_Scan_Location* - Scan location
- *Barratt_Barratt_P1_Edu* - Parent 1 level of education
- *Barratt_Barratt_P1_Occ* - Parent 1 occupation
- *Barratt_Barratt_P2_Edu* - Parent 2 level of education
- *Barratt_Barratt_P2_Occ* - Parent 2 occupation

3.3 Clinical Assessments Data

This dataset comprises 23,047 observations, including symptom ratings, diagnostic outcomes, and subscale scores from standardized psychiatric assessments. These features help quantify the severity and profile of ADHD-related behaviors across individuals.

The variables include the following:

- *EHQ_EHQ_Total* - Edinburgh Handedness Questionnaire Laterality Index (Score)
- *ColorVision_CV_Score* - Ishihara Color Vision Test Color vision test score
- *APQ_P_APQ_P_CP* - Alabama Parenting Questionnaire - Parent Report -Corporal Punishment Score
- *APQ_P_APQ_P_ID* - Alabama Parenting Questionnaire - Parent Report - Inconsistent Discipline Score
- *APQ_P_APQ_P_INV* - Alabama Parenting Questionnaire - Parent Report - Involvement Score
- *APQ_P_APQ_P_OPD* - Alabama Parenting Questionnaire - Parent Report - Other Discipline Practices Score
- *APQ_P_APQ_P_PM* - Alabama Parenting Questionnaire - Parent Report - Poor Monitoring/Supervision Score
- *APQ_P_APQ_P_PP* - Alabama Parenting Questionnaire - Parent Report - Positive Parenting Score
- *SDQ_SDQ_Conduct_Problems* - Strength and Difficulties Questionnaire - Conduct problems scale
- *SDQ_SDQ_Difficulties_Total* - Strength and Difficulties Questionnaire - Total Difficulties Score
- *SDQ_SDQ_Emotional_Problems* - Strength and Difficulties Questionnaire - Emotional Problems Scale
- *SDQ_SDQ_Externalizing* - Strength and Difficulties Questionnaire - Externalizing Score
- *SDQ_SDQ_Generating_Impact* - Strength and Difficulties Questionnaire - Generating Impact Scores
- *SDQ_SDQ_Hyperactivity* - Strength and Difficulties Questionnaire - Hyperactivity Scale
- *SDQ_SDQ_Internalizing* - Strength and Difficulties Questionnaire - Internalizing Score
- *SDQ_SDQ_Peer_Problems* - Strength and Difficulties Questionnaire - Peer Problems Scale
- *SDQ_SDQ_Prosocial* - Strength and Difficulties Questionnaire - Prosocial Scale
- *MRI_Track,Age_at_Scan* - MRI Information - Age at time of MRI scan

4 Data Preprocessing and Dimensionality Reduction

To ensure consistency and enhance the utility of our predictive models, the data processing in this project includes data cleaning, transformation, and dimensionality reduction.

4.1 Data Integration

The three datasets were merged on a common participant identifier (*participant_id*) to form a single comprehensive dataset. This enables us to analyze interactions between brain connectivity, clinical symptoms, and demographic factors concerning ADHD diagnosis and sex differences.

4.2 Handling Missing Data

Missing values were addressed according to the type of feature.

Because most of the quantitative features are skewed, we utilize median imputation.

For categorical features, mode imputation was applied to preserve the categorical distribution and mitigate the introduction of noise.

4.3 Outlier Detection and Treatment

To mitigate the influence of anomalous values, the Interquartile Range (IQR) method was used. Outliers were defined as data points lying beyond 1.5×IQR from the first (Q1) or third quartile (Q3).

The effectiveness of the outlier removal process is visually confirmed in the box plots shown in Figure 1. The plot on the left shows the original distribution of selected features, where numerous outliers are indicated as individual points beyond the whiskers. After applying the IQR-based filtering (right plot), these extreme values were successfully removed, resulting in more compact and normally distributed feature ranges. This step enhanced the overall homogeneity of the data and ensured that the modeling phase would be less sensitive to noise and distortion from anomalous values.

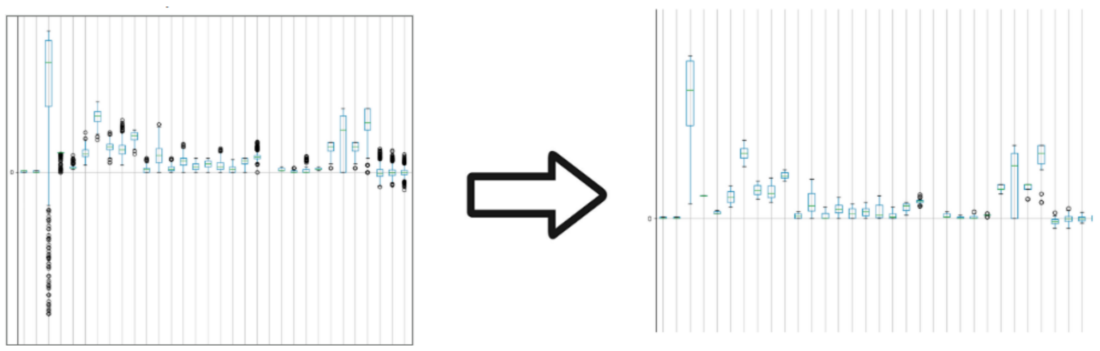


Figure 2: boxplot before and after handling outliers

4.4 Feature Engineering for Functional Connectivity

We extracted functional connectivity matrices with a dimensionality of $200 * 200$, representing pairwise connections among 200 brain regions. This whole-brain parcellation divides the brain into 200 regions of interest (ROIs), resulting in a symmetric 200×200 correlation matrix for each participant. Because the matrix is symmetric about the diagonal (each connection between region i and j is equivalent to j–i), we focused on the upper triangular elements. This yields a total of 19,900 unique pairwise connections (since $200 \times 199/2 = 19{,}900$) as the features describing each individual's functional connectome. In other words, each subject is characterized by a 19,900-dimensional feature vector representing all unique functional connectivity edges across the 200 ROIs.

To prepare these features for analysis, we flattened the upper triangle of each subject's connectivity matrix into a one-dimensional vector, representing the subject's neural connectome profile. This process yielded a very high-dimensional feature space, requiring further steps to enhance tractability and modeling relevance.

To facilitate neuroscientifically meaningful interpretation, we computed three classes of graph summary features from the raw edge values:

- Global connectivity strength: the mean Pearson correlation across all edges, indicative of overall brain synchrony.
- Network sparsity: the proportion of connections below a minimal threshold, used to infer general network efficiency.
- Regional connectivity aggregates: average connectivity strengths for functionally grouped regions (e.g., frontal, parietal, occipital), enabling interpretation of spatially localized neural dynamics.

4.5 Dimensionality Reduction with PCA

The expansion to 19,900 connectivity features represents a substantially higher-dimensional feature space. This high dimensionality poses challenges for statistical modeling and increases the risk of overfitting. To address this, we applied principal component analysis (PCA). The input to PCA is the 19,900-dimensional connectivity feature vector for each subject. We elected to retain the top K principal components (accounting for 90% of variance) as the reduced feature set for subsequent analyses. As a result, we balance the need to reduce dimensionality and to preserve most of the information present in the original connectivity data.

4.6 Encoding Categorical Variables

Categorical variables such as *Barratt_Barratt_P1_Occ* and *PreInt_Demos_Fam_Child_Race* were converted into binary features using one-hot encoding.

4.7 Correlation Analysis and Feature Selection

To reduce redundancy and improve model efficiency, we applied correlation-based feature selection to the dataset. We began by isolating all numerical variables and computing their absolute Pearson correlation matrix, which quantifies the strength of linear associations between feature pairs.

We then scanned for features that has a correlation coefficient greater than 0.8 with any other feature, which may indicate strong collinearity and possible duplication of information.

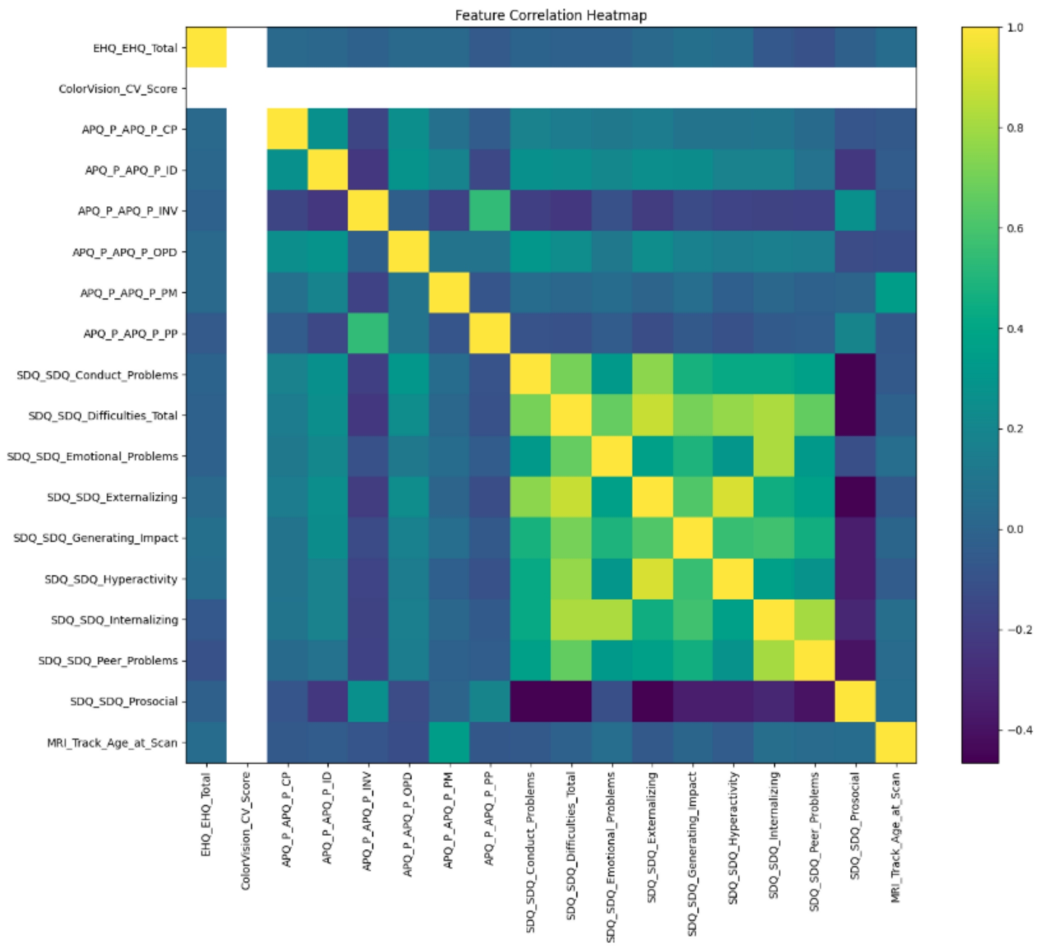


Figure 3: Correlation Matrix

The Figure 3 above is the heatmap of the absolute correlation matrix. The yellow diagonal indicates self-correlation, while the darker shades represent stronger off-diagonal correlations among the variables. We can see clusters of correlated features within the SDQ subscales.

As a result of this process, four features were identified as highly correlated and subsequently removed from the dataset:

SDQ_SDQ_Externalizing
SDQ_SDQ_Hyperactivity
SDQ_SDQ_Internalizing
SDQ_SDQ_Peer_Problems

5 Exploratory Data Analysis

5.1 Clinical Assessment Features and ADHD Diagnosis

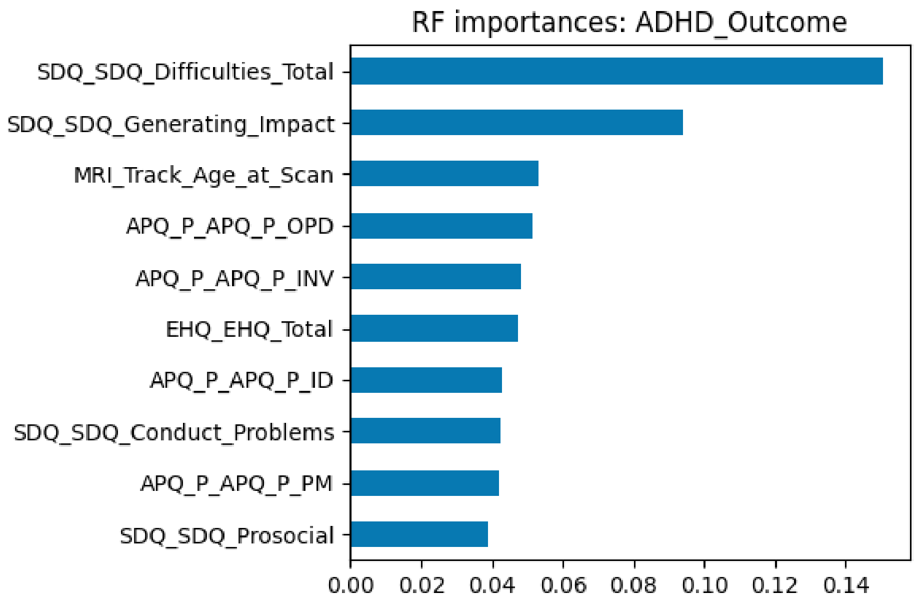


Figure 4: Clinical Assessment Features and ADHD Diagnosis

According to Figure 4, the Random Forest model for ADHD classification highlighted *SDQ Total Difficulties* as the single most important feature, with *SDQ Impact* as the second-highest contributor. These two features capture the severity of a child's behavioral problems and their resulting impairment. Both of these SDQ measures ranked high suggests that children diagnosed with ADHD in the sample had markedly higher problem scores and associated functional impairment compared to their non-ADHD peers. This aligns with our expectations, as ADHD is characterized by elevated behavioral issues and significant impact on functioning. The prominence of these two related features also implies some redundancy: children with high total difficulty scores usually also have high impact scores, since greater symptom severity tends to cause greater impairment.

In addition to the *SDQ*, a few parenting questionnaire showed predictive value for ADHD. "Other Discipline Practices" (*APQ_OPD*) and "Parental Involvement" (*APQ_INV*) are among the top five list. Therefore, parents of children with ADHD may resort to a greater variety of discipline strategies or have changed the levels of participation in the child's activities.

In summary, children with ADHD show higher overall difficulties and impairment and come from family environments marked by different parenting dynamics.

5.2 Clinical Assessment Features and Sex Differences

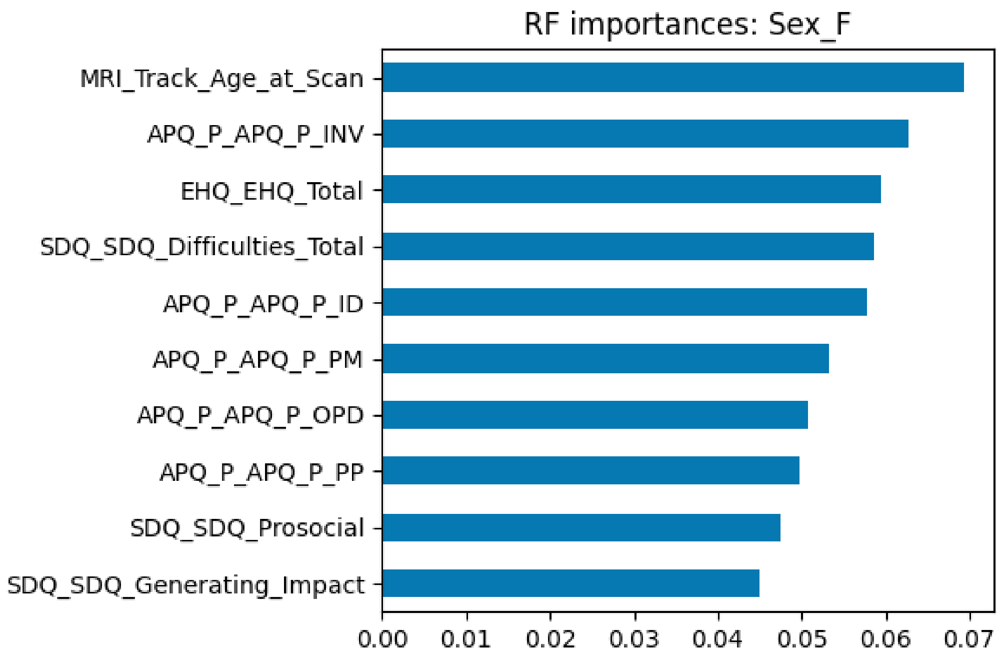


Figure 5: Clinical Assessment Features and Sex Differences

When predicting biological sex (Sex_F) from the clinical questionnaire set, the Random Forest model picked up *Age at MRI Scan* and "parental Involvement" (*APQ_INV*) as the top two.

The prominence of age suggests there may have been an age imbalance or age-related difference between female and male participants in the sample. Female participants were on average slightly older at the time of scan than the males, which may indicate that female may get their ADHD diagnosed older than male.

The *APQ_INV* (parent Involvement score) being a strong predictor indicates that parenting behavior differed by the child's sex: higher parental involvement scores were associated with the child being female.

5.3 fMRI Network Features and ADHD Diagnosis

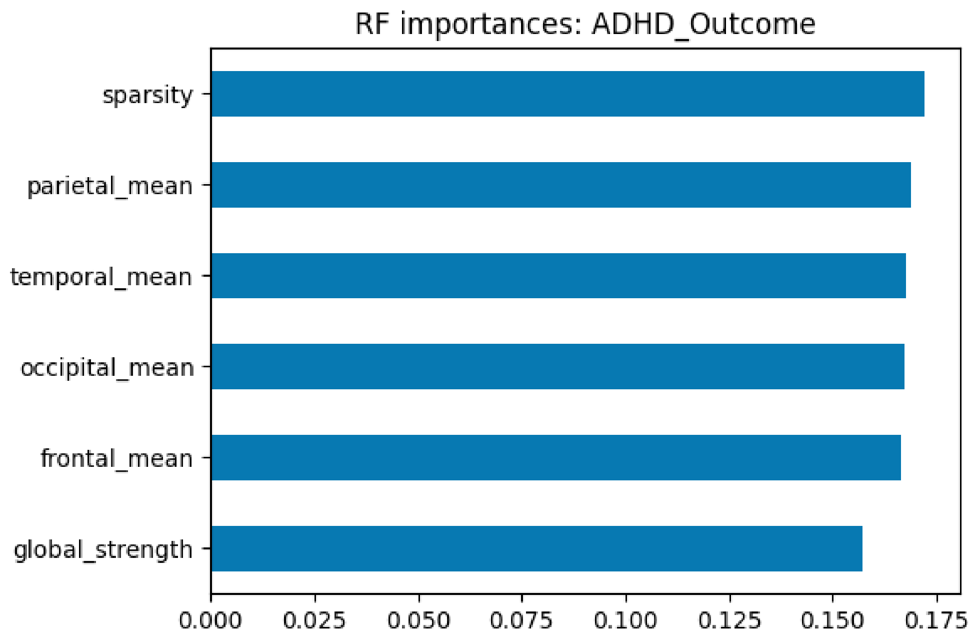


Figure 6: fMRI and ADHD

According to Figure 6, the Random Forest model for ADHD outcome found that *network sparsity* was the most important feature, with *parietal_mean connectivity* a close second.

The high ranking of *network sparsity* implies that ADHD brains differed in how densely connected or diffuse their functional networks are. The importance of the *parietal_mean connectivity* points that connectivity involving parietal lobe regions was particularly discriminative between ADHD and non-ADHD. The parietal lobe is heavily involved in attentional networks and executive function (e.g., the dorsal attention network and frontoparietal control systems), which are known to function atypically in ADHD.

5.4 fMRI Network Features and Sex Differences

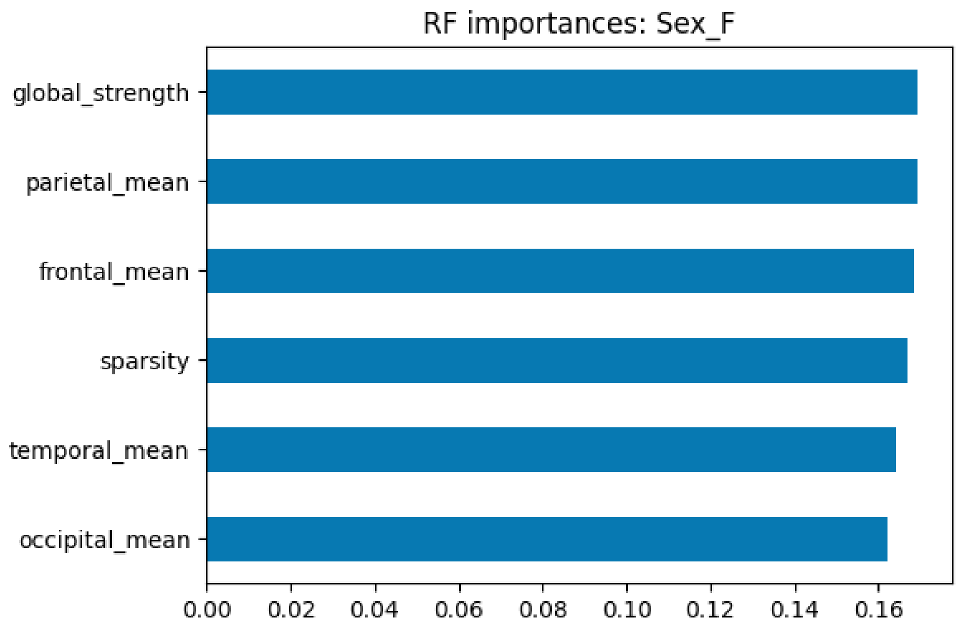


Figure 7: fMRI with sex differences

According to Figure 7, the Random Forest classifier for *Sex-F* found *Global connectivity strength* and *parietal_mean connectivity* as the top predictors, with *frontal_mean*, *network sparsity*, and *temporal_mean* very close behind. This pattern suggests that sex differences in brain connectivity were subtle and distributed rather than localized.

6 Methodology and analysis procedure

We investigated five supervised learning models—Logistic Regression, Random Forest, Gradient-Boosted Trees (XGBoost and LightGBM), and a Multi-Layer Perceptron (MLP) neural network—to predict ADHD status and sex. The full dataset was split into a training set (80%) and a hold-out testing set (20%), stratified by class labels.

Our model development strategy was iterative and comparative. We evaluated each model under three feature-set configurations:

1. **Meta-only(baseline):** demographic and clinical assessment features only.
2. **Meta+FC:** Meta features plus PCA-derived functional connectivity components.
3. **Feature-Engineered:** On top of (2), we applied systematic feature engineering and selection, in particular:

- *Binning continuous scores* (e.g., discretizing questionnaire totals into low/medium/high categories),
- *Polynomial and interaction terms* (e.g., crossing age with symptom severity),
- *SelectKBest* univariate filtering to retain the top 20 features per task,
- *Recursive feature elimination* guided by model importance metrics.

Model performance was assessed primarily with the Area Under the ROC Curve (AUC), which provides a threshold-independent measure of discriminatory power under class imbalance. We then applied *5-fold cross-validation* on the training set—verifying that observed AUC gains generalized beyond a single train-test split to guard against overfitting and ensure robustness. Finally, based on these experimental results, we adjusted whether to train distinct models for each task (ADHD vs. sex) to allow task-specific optimization.

7 Results and Discovery for different models

7.1 Logistic Regression Model

We implemented three successive logistic regression models (Meta-only, Meta+FC, and Feature-Engineered) to incrementally enrich the feature set and evaluate their impact on predicting ADHD status and sex. All models were trained under identical conditions (an 80/20 train–test split and the regularization settings from Section 6) and evaluated primarily by the area under the ROC curve (AUC). This staged modeling strategy, starting from a simple baseline and increasing complexity, allows us to isolate the contribution of neuroimaging features and targeted feature engineering to overall performance and interpretability.

Table 1: AUC Comparison of Logistic Regression

Pipeline	**Sex AUC**	**ADHD AUC**
Meta-only (baseline)	0.570	0.874
Meta + FC (raw concatenation)	0.615	0.668
Multi-task w/ separate selection	0.658	0.870

- **Meta-only baseline**

The first logistic regression model was trained using only the metadata features, consisting of 71 demographic and clinical variables (e.g., age, questionnaire scores, and cognitive test results). This meta-only configuration served as a performance baseline that relied exclusively on non-imaging predictors. On the held-out test set, the model achieved an AUC of approximately **0.874** for ADHD diagnosis, indicating that metadata alone carried substantial discriminative signal for this task. In contrast, the model's AUC for sex classification was only about **0.57**.

This suggests that demographic and behavioral features contained limited information relevant to participant sex, perhaps because few meta features were sex-specific and the sex classes were imbalanced. The baseline experiment thus established a reference level of accuracy using readily available features. Having quantified what metadata alone can achieve, we next explored whether incorporating brain connectivity measures could improve the model, especially for the more challenging sex prediction task.

- **Meta+FC**

In the second phase, we extended the feature set by incorporating functional connectivity (FC) components derived from fMRI data. To reduce dimensionality, principal component analysis (PCA) was applied to each participant's connectivity matrix, and the top components were appended to the 71 meta features, yielding a combined input of approximately 700 dimensions. The intent was to capture additional neural patterns potentially relevant to ADHD or sex classification.

The resulting model showed a mixed outcome. For sex prediction, the test-set AUC increased to **0.615** (from **0.57** in the baseline), suggesting that some FC-derived features provided relevant information. However, ADHD performance declined, with the AUC dropping to **0.668** (from **0.874**), indicating that the added features introduced noise and diluted the predictive signal from the meta-data.

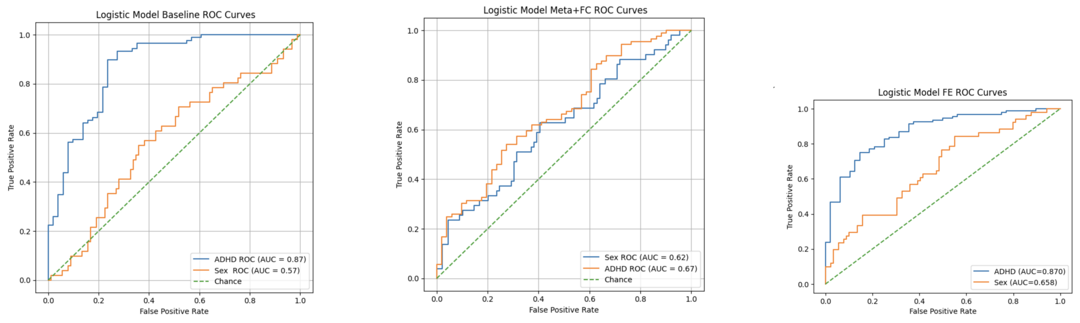
This degradation highlights the risk of incorporating high-dimensional features without filtering. While FC features may improve certain tasks (e.g., sex classification), unfiltered inclusion can hinder performance, particularly for ADHD. These results motivated a more selective feature engineering approach in the next phase.

- **Feature-engineered**

In the final phase, we refined the input space by selectively engineering and filtering features based on insights from the previous model. Instead of retaining all FC components, we returned to the meta-only feature set and augmented it with a small subset of connectivity-derived variables identified as potentially informative. We further applied task-specific feature transformations, including discretization, interaction terms, and univariate selection, resulting in compact input sets for each classification task.

This targeted approach led to improved and more balanced performance. For ADHD classification, the AUC returned to **0.87**, nearly matching the baseline, while sex classification improved further to approximately **0.66**. These results indicate that selective inclusion of well-chosen features can preserve model simplicity while enhancing discriminative power.

Compared to the prior phases, this model offered a better trade-off between complexity and performance, demonstrating the value of domain-guided feature construction in high-dimensional predictive tasks.



(a) Meta-only Baseline (b) Meta + FC (c) Feature Engineering

Figure 8: Comparison of Logistics Sex/ADHD Model ROC Graph

7.2 Random Forest Model

To evaluate the Random Forest (RF) classifier's performance, we implemented a three-stage pipeline progressing from a meta-data baseline to an augmented model with fMRI connectivity features, and finally to a cross-validated assessment of generalizability. Each stage was evaluated on two tasks—ADHD diagnosis and sex prediction—using AUC on held-out data.

Table 2: Comparison of Random Forest Pipelines on Sex and ADHD AUCs

Pipeline	**Sex CV AUC**	**Sex Hold-out AUC**	**ADHD CV AUC**	**ADHD Hold-out AUC**
Baseline RF (all Meta+FC)	—	0.592	—	0.803
Feature-Engineered + RF	0.589	0.752	0.793	0.850

- **Meta-only (baseline)**

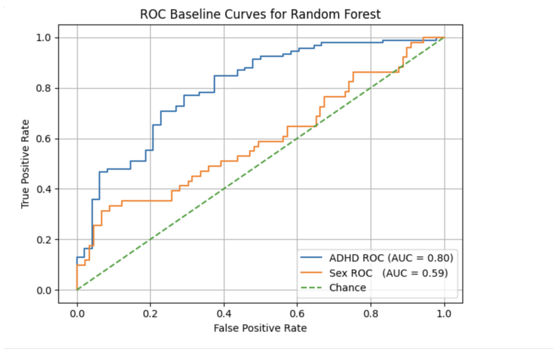
In the first stage, we trained a Random-Forest (RF) model using the 71 meta features (demographic, clinical, and questionnaire variables), without any feature engineering or neuroimaging inputs.

These numbers show that meta-data alone still carry substantial signal for ADHD (AUC 0.80), but provide limited discriminatory power for sex (AUC 0.59, only modestly above chance). Compared with the logistic baseline (Sex AUC 0.57, ADHD AUC 0.874), the RF extracts slightly more non-linear structure for sex yet underperforms on ADHD, suggesting that tree-based ensembles need richer feature representations to rival the linear model on the ADHD task. Overall, the baseline confirms that additional feature engineering or connectivity information is essential—especially for stable sex classification—while meta features alone are already reasonably informative for ADHD.

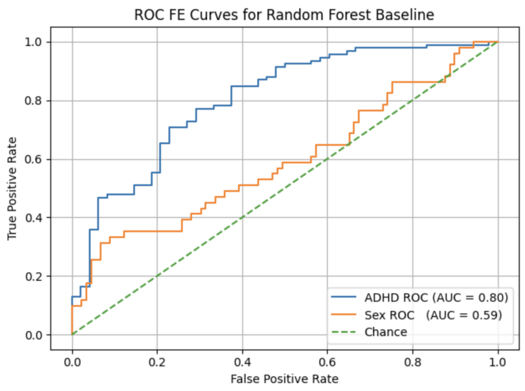
- **Feature Engineering+FC**

The second stage expanded the predictor set by incorporating neuroimaging data in the form of fMRI functional connectivity (FC) features, alongside the meta-data. High-dimensional FC matrices were first reduced via principal component analysis (PCA) to obtain a manageable number of summary components (see Section 6 for details of the PCA reduction and feature selection methods). These PCA-derived connectivity features were then appended to the 71 meta features, and a feature selection step was applied to filter out noisy or redundant variables and retain the most informative predictors for each outcome. The RF was retrained on this enriched, pruned feature set containing both meta features and selected connectivity features.

On the hold-out test, adding selected connectivity features improved both tasks: ADHD AUC rose to **0.85**, while sex AUC increased markedly to **0.752**. This highlights the substantial sex-related signal captured by engineered connectivity features, which outperformed logistic regression (AUC up to **0.658** for sex). Unlike the logistic model, which lost ADHD accuracy when all raw FC features were included, RF preserved ADHD performance and achieved a significant gain in sex prediction through targeted feature selection.



(a) Baseline RF



(b) Feature-Engineered RF

Figure 9: Comparison of Random Forest ROC curves on sex classification.

- **Cross-Validation**
  The third stage assessed the robustness and generalizability of the optimized RF model by evaluating it under cross-validation. Using the final feature set from Stage 2 (the meta features plus the selected PCA connectivity features), we performed a 5-fold cross-validation within the training data, measuring the model's average performance on held-out folds. This provides a more rigorous estimate of generalization performance, mitigating the risk that the single train/test split in earlier stages was overly favorable or that feature selection had overfit to idiosyncrasies of that particular split.
  Cross-validation revealed that RF's ADHD performance drop to 0.793, and to be worse, sex classification AUC dropped sharply from **~0.752** (hold-out) to **~0.589** (CV), nearly reverting to the logistic baseline and indicating clear overfitting. This suggests that the apparent sex-prediction improvement was largely split-specific and did not generalize across folds.

- **Next Step**
  The drop in sex–AUC from **~0.75** (hold-out) to **~0.59** (5-fold CV) reveals that the RF overfit the hold-out split for sex prediction. To address this and explore a model less prone to split-specific overfitting, we will next develop a Multi-Layer Perceptron (MLP). As with RF, the MLP pipeline will incorporate the same engineered feature set, employ rigorous cross-validation during both feature selection and hyperparameter tuning, and include early-stopping and regularization to safeguard against overfitting while preserving the strong ADHD performance.

7.3 Multi-Layer Perceptron (MLP) Model Pipelines

In this section, we evaluate a Multi-Layer Perceptron (MLP) with a single hidden layer of 100 neurons (`max_iter=500`) for ADHD diagnosis and sex classification. The train–test split follows the same stratified 80/20 scheme as before, and performance is assessed via 5-fold cross-validation. We test three feature pipelines: meta-data only, PCA fusion, and intersection/union feature subsets.

Table 3: MLP Pipelines: Sex and ADHD AUC Comparison

Pipeline	**Sex CV AUC**	**Sex Hold-out AUC**	**ADHD CV AUC**	**ADHD Hold-out AUC**
Baseline MLP	0.579	0.540	0.730	0.793
PCA + MLP	0.611	0.531	0.709	0.736
FE + MLP	0.548	0.452	0.784	0.810

- **Baseline MLP**
  This pipeline uses only the encoded meta-data (71 demographic and clinical features). The network architecture and preprocessing mirror those of the earlier random forest baseline.

  Results:

  - Sex classification: 5-fold CV AUC ≈ **0.579**, hold-out AUC ≈ **0.540**.
  - ADHD prediction: 5-fold CV AUC ≈ **0.730**, hold-out AUC ≈ **0.793**.

  Meta-data alone contain sufficient signal for ADHD but almost none for sex, reproducing the trends seen in logistic regression and random forest baselines; even when RF raised sex AUC to ~ 0.75, the MLP remained at ~ 0.54, indicating it cannot learn the nonlinear sex-related patterns from raw meta-data. As Figure1 shows, the baseline MLP's sex ROC curve nearly coincides with chance, while its ADHD performance is stable and comparable to RF's hold-out AUC ≈ 0.836.

- **PCA + MLP**
  In the second stage, we apply PCA to the functional-connectivity matrix to extract the top 50 principal components, then concatenate these with the meta-data and train the same MLP architecture.
  Results:

  - Sex classification: CV AUC ≈ **0.611**, hold-out AUC ≈ **0.531**.
  - ADHD prediction: CV AUC ≈ **0.709**, hold-out AUC ≈ **0.736**.

  PCA-derived features substantially boost sex classification, confirming that FC data contain sex-related variance; the gain for ADHD is marginal and exhibits diminishing returns. The drop from CV to hold-out (0.611→0.531) indicates mild overfitting.

- **Feature Engineering+MLP**
  Using the same engineered-feature pipeline (quantile binning, interaction terms, polynomial expansions) and univariate SelectKBest from the RF analysis, we train the identical MLP on the top-20 selected features.
  Results:

  - Sex classification: CV AUC ≈ **0.548**, hold-out AUC ≈ **0.452**.
  - ADHD prediction: CV AUC ≈ **0.784**, hold-out AUC ≈ **0.810**.

  Engineered features cause MLP sex AUC to drop from 0.548 (CV) to 0.452 (hold-out), revealing poor generalization and amplified noise. In contrast, ADHD AUC rises from 0.784

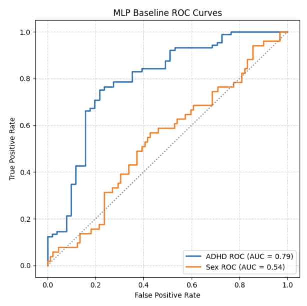
to 0.810, showing reliable signal gains. Thus, while manual feature pruning benefits strong ADHD signals, sex classification demands stronger regularization or more powerful models to generalize beyond training artifacts.

- **Intersection/Union + MLP**
  Finally, we form two candidate feature subsets: the intersection (7 features) and the union (33 features) of the top-20 lists from random forest importance and univariate selection. We train the same MLP on each subset.
  Results:

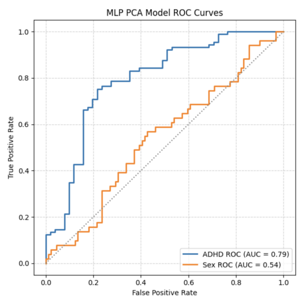
  - Intersection: CV AUC $\approx$ **0.599** $\rightarrow$ hold-out AUC $\approx$ **0.554**.
  - Union: CV AUC $\approx$ **0.571** $\rightarrow$ hold-out AUC $\approx$ **0.467**.

  The intersection strategy yields only a slight CV gain but reverts to baseline performance on the hold-out set, indicating high variance and overfitting; the union strategy suffers from feature redundancy, causing a sharp performance drop. ADHD performance remains stable, having plateaued in the previous stage.

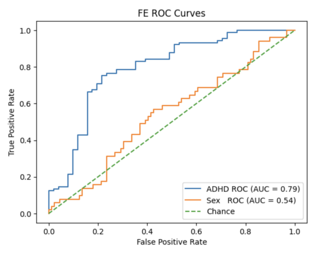
Next, we will introduce LightGBM, leveraging its built-in regularization and feature subsampling to model higher-order nonlinear interactions across the full feature set, aiming for more stable and reliable classification performance on both sex and ADHD tasks.



(a) Baseline MLP



(b) PCA + MLP



(c) FE + MLP

Figure 10: Comparison of MLP Sex/ADHD Model ROC Graph

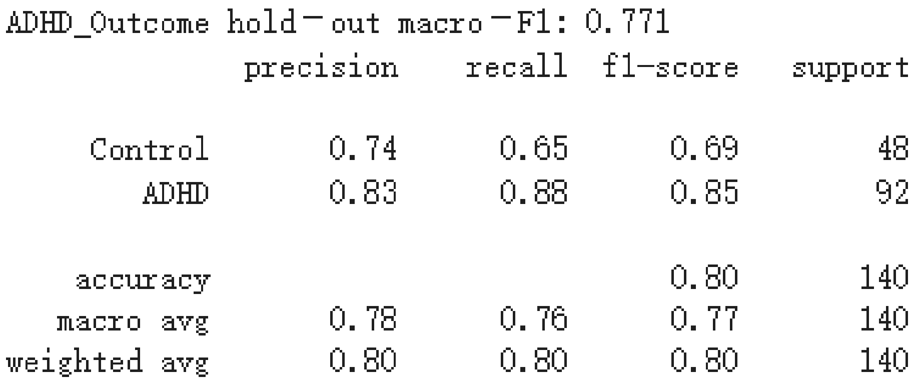
7.4 Gradient-Boosted Trees

In this section, we trained ensemble tree-based models to predict ADHD diagnosis and sex classification using different combinations of clinical and neuroimaging features. For ADHD classification, we employed XGBoost; for Sex classification, we used LightGBM.

Model performance is assessed with Macro-F1 score and overall accuracy. Macro-F1 is the unweighted average of the F1-scores for each class (ADHD vs non-ADHD; Female vs Male). Given the potential class imbalance (especially in *Sex_F*, where one sex may be under-represented), macro-F1 provides a more balanced evaluation than accuracy alone.

7.4.1 ADHD Prediction

- Baseline XGBoost

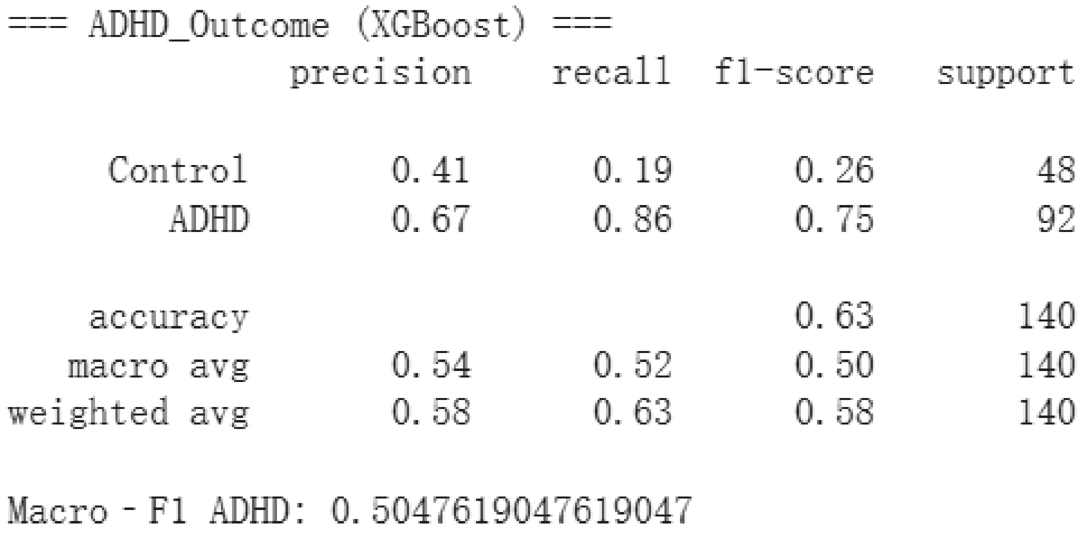


```
ADHD_Outcome hold－out macro－F1: 0.771
              precision    recall  f1-score   support

     Control       0.74      0.65      0.69        48
        ADHD       0.83      0.88      0.85        92

    accuracy                           0.80       140
   macro avg       0.78      0.76      0.77       140
weighted avg       0.80      0.80      0.80       140
```

Figure 11: Baseline XGBoost

Using only the encoded metadata ((71 demographic and clinical features), the XGBoost classifier achieved a Macro-F1 of 0.771 with 80% accuracy for ADHD prediction. This suggests that the clinical and questionnaire measures contain strong signals related to ADHD diagnosis. Both precision and recall were high for each class (ADHD and typically-developing), the model did not overly favor one class, implying it could detect ADHD-positive cases with nearly the same proficiency as controls.

- Handcrafted fMRI Features XGBoost

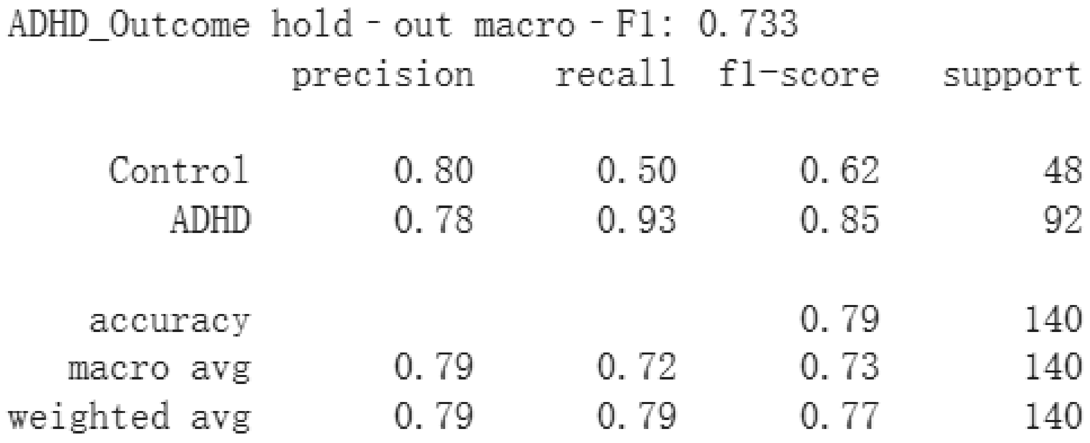


```
=== ADHD_Outcome (XGBoost) ===
              precision    recall  f1-score   support

     Control       0.41      0.19      0.26        48
        ADHD       0.67      0.86      0.75        92

    accuracy                           0.63       140
   macro avg       0.54      0.52      0.50       140
weighted avg       0.58      0.63      0.58       140

Macro－F1 ADHD: 0.5047619047619047
```

Figure 12: Handcrafted XGBoost

When using the meta plus handcrafted functional connectivity features (global strength, sparsity, regional mean connectivities), the ADHD classification performance dropped substantially. The XGBoost model in this setting obtained a Macro-F1 of 0.504 and an accuracy of 0.63. This indicates that, with just a few summary graph metrics, the model struggled to differentiate ADHD from non-ADHD. The precision-recall breakdown suggested the classifier was often defaulting to predicting the majority class. The loss in performance relative to the clinical baseline highlights that these coarse graph features did not capture enough discriminative information. While interpretable, they likely omit nuanced patterns in the full connectivity data that are important for classification. This result underscores a trade-off: by using only a small set of interpretable connectivity features, we gained explainability but at the cost of predictive power.

- Meta + PCA fMRI Features XGBoost



```
ADHD_Outcome hold-out macro-F1: 0.733
              precision    recall  f1-score   support

     Control       0.80      0.50      0.62        48
        ADHD       0.78      0.93      0.85        92

    accuracy                           0.79       140
   macro avg       0.79      0.72      0.73       140
weighted avg       0.79      0.79      0.77       140
```

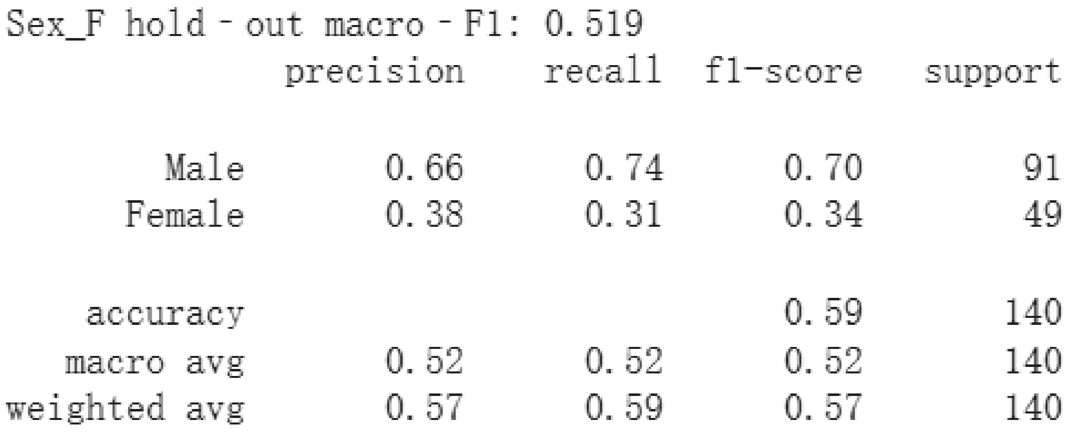
Figure 13: Meta+Func

Using Meta plus PCA-transformed functional features yielded much better performance than the handcrafted features. The XGBoost model using the top principal components achieved Macro-F1 = 0.733 and accuracy = 0.79 for ADHD. This suggests that the rich information in the full connectivity matrices does have predictive value for ADHD status. The model's precision and recall for ADHD vs control were more balanced than with the handcrafted features, demonstrating that it could identify ADHD cases more successfully.

In summary, for ADHD, the baseline clinical model remained the top performer, but the PCA-based functional model demonstrated that much of the relevant signal from fMRI could be captured when enough components were included, whereas overly summarized connectivity features were insufficient.

7.4.2 Sex Classification

- Baseline LightGBM

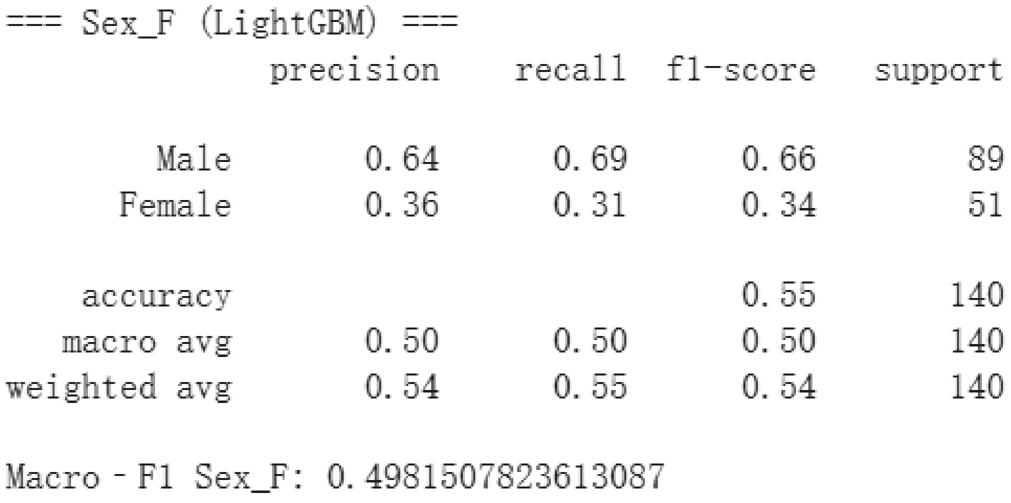


```
Sex_F hold－out macro－F1: 0.519
              precision    recall  f1-score   support

        Male       0.66      0.74      0.70        91
      Female       0.38      0.31      0.34        49

    accuracy                           0.59       140
   macro avg       0.52      0.52      0.52       140
weighted avg       0.57      0.59      0.57       140
```

Figure 14: Baseline LightGBM

The LightGBM model using metadata alone had a Macro-F1 of 0.519 and accuracy of 0.59 in predicting biological sex (female vs male). This indicates only modest predictive power from the available clinical variables for distinguishing sex. An accuracy of 59% is only slightly above chance (50%), and the macro-F1 of 0.52 suggests that the model was only marginally balanced in its predictions. Class-wise recall revealed that the model tended to predict the majority sex more frequently – higher recall for the Male class than the Female class.

- Handcrafted fMRI Features LightGBM

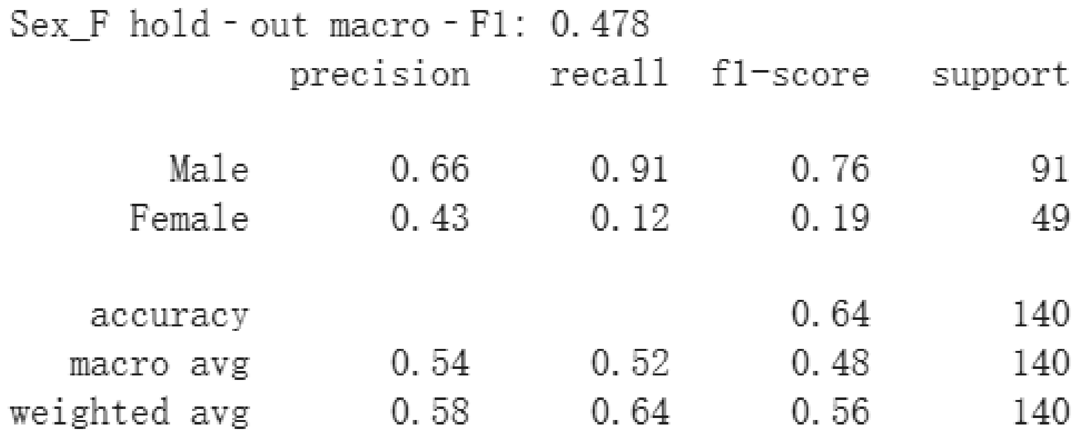


```
=== Sex_F (LightGBM) ===
              precision    recall  f1-score   support

        Male       0.64      0.69      0.66        89
      Female       0.36      0.31      0.34        51

    accuracy                           0.55       140
   macro avg       0.50      0.50      0.50       140
weighted avg       0.54      0.55      0.54       140

Macro－F1 Sex_F: 0.4981507823613087
```

Figure 15: Handcrafted Sex classification

Using the Random-Forest-selected connectivity metrics (global strength, sparsity, regional means) for sex classification yielded Macro-F1 = 0.498 and accuracy = 0.55. This is at chance-level for macro-F1 (0.50) and slightly above chance for accuracy. The model learned very little sex-distinguishing pattern from those features. In practice, it was often predicting the majority class (Male) to achieve 55% accuracy, while barely identifying Female subjects (Female recall was very low, on the order of the class prior). This indicates that summary measures like overall connectivity strength or sparsity do not markedly differ between male and female brains in this age group, or at least not enough to be captured by a simple classifier. Sex differences in functional connectivity may be subtle and distributed across many connections, which a few global metrics cannot capture. The poor performance reinforces the need for higher-dimensional or more complex features to detect sex differences in brain connectivity.

- Meta+PCA fMRI Features LightGBM



```
Sex_F hold - out macro - F1: 0.478
              precision    recall  f1-score   support

        Male       0.66      0.91      0.76        91
      Female       0.43      0.12      0.19        49

    accuracy                           0.64       140
   macro avg       0.54      0.52      0.48       140
weighted avg       0.58      0.64      0.56       140
```

Figure 16: Meta+PCA LightGBM

The Meta plus PCA-based functional connectivity LightGBM for sex achieved the highest accuracy of the three models (64%), but its Macro-F1 was the lowest (0.478). In this model, the inclusion of many principal components from the connectivity data gave it some ability to correctly classify males versus females, indicating the model found more sex-related signal than the Meta model did. However, the drop in macro-F1 shows a growing imbalance in predictive performance. The model was strongly biased toward predicting the majority class (Male), correctly labeling many males but misclassifying most females. The precision for the female class was also poor due to the scarcity of female predictions.

This outcome highlights a classic precision–recall trade-off under imbalance: the model tuned for raw accuracy sacrificed sensitivity to the minority class.

8 Discussion and Future Extension

In conclusion, this report demonstrates that clinical and behavioral assessment data serve as robust predictors of ADHD in youth, often outperforming complex neuroimaging features. Our models achieved strong accuracy in ADHD classification, primarily driven by questionnaire indices of behavior and symptoms, reinforcing that such ratings (e.g., SDQ Total Difficulties) encapsulate much of the diagnostic signal.

Adding resting-state fMRI connectivity features yielded only minor improvements for ADHD prediction, demonstrating that current neural measures (especially in aggregate forms like global network metrics or PCA components) provide limited incremental value beyond traditional clinical metrics. By contrast, the consistent difficulty in predicting sex from the same multimodal data highlights a fundamental challenge: the functional brain connectivity differences between male and female adolescents are either very subtle or not captured by the features and models used. All tested classifiers struggled to distinguish sex, frequently overfitting or predicting only the majority sex class, which suggests that the available feature space did not contain strong sex-discriminative patterns.

Looking ahead, our findings point to future research. First, more granular characterization of functional connectivity should be explored. Instead of coarse summaries or broad principal components, analyses could focus on specific brain network connections or dynamic connectivity patterns that might reveal subtle sex differences that were obscured in this study. Second, it may be helpful to model sex as a moderating factor in ADHD-related analyses. For example, building sex-specific ADHD classifiers or including interaction terms between sex and connectivity features. Third, the use of larger datasets and advanced machine learning techniques (such as deep learning models capable of handling high-dimensional fMRI data) could enhance generalization and help avoid the overfitting seen in the sex classification task. Ultimately, integrating clinical and neural data in more nuanced ways will be important for future studies: while clinical assessments are effective for predicting ADHD outcomes, uncovering how brain connectivity contributes to these outcomes (and differs by sex) may require methods that can manage complex interaction effects. Continued research along these lines will improve our understanding of when neuroimaging biomarkers add value to clinical diagnosis and how sex differences in the brain relate to neurodevelopmental disorders.

9 Appendices

```
## Data Preprocessing

#load the data
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
drive.mount('/content/drive')
import pickle
import os
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
import numpy as np
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import HistGradientBoostingClassifier

train_sol = pd.read_excel('/content/drive/MyDrive/Datathon/TRAIN_NEW/TRAINING_SOLUTIONS.xlsx')
train_sol.head()

train_cat = pd.read_excel('/content/drive/MyDrive/Datathon/TRAIN_NEW/TRAIN_CATEGORICAL_METADATA_new.
train_cat.head()

train_fun = pd.read_csv('/content/drive/MyDrive/Datathon/TRAIN_NEW/TRAIN_FUNCTIONAL_CONNECTOME_MATRI
train_fun.head()

train_qun = pd.read_excel('/content/drive/MyDrive/Datathon/TRAIN_NEW/TRAIN_QUANTITATIVE_METADATA_new
train_qun.head()

#check if duplicated
for df,name in [(train_qun,'quant'),(train_cat,'cat'),(train_fun,'fc')]:
    dups = df['participant_id'].duplicated().sum()
    print(f'{name}: {dups} duplicate ids')

# check if all IDs line up with labels
for df,name in [(train_qun,'quant'),(train_cat,'cat'),(train_fun,'fc')]:
    diff = set(train_sol.participant_id) - set(df.participant_id)
    print(f'{name} missing {len(diff)} label ids')
#target distribution
for col in ['ADHD_Outcome','Sex_F']:
    counts = train_sol[col].value_counts().sort_index()
    ax = counts.plot(kind='bar', title=f'{col} class balance')
    plt.show()
    print(counts, '\n')

train_meta = (train_sol
        .merge(train_qun, on='participant_id', how='left')
        .merge(train_cat, on='participant_id', how='left'))


test_meta = (test_qun
        .merge(test_cat, on='participant_id', how='left'))

train_meta.head()

#NA
miss_tbl = (meta
            .isna()
            .mean()
            .mul(100)
            .sort_values(ascending=False))

print(miss_tbl.head(20))
print('Cols with >80 % NA:', (miss_tbl>80).sum())

#convert column type
train_meta['Basic_Demos_Study_Site'] = train_meta['Basic_Demos_Study_Site'].astype('category')
train_meta['PreInt_Demos_Fam_Child_Ethnicity'] = train_meta['PreInt_Demos_Fam_Child_Ethnicity'].asty
train_meta['PreInt_Demos_Fam_Child_Race'] = train_meta['PreInt_Demos_Fam_Child_Race'].astype('catego
train_meta['MRI_Track_Scan_Location'] = train_meta['MRI_Track_Scan_Location'].astype('category')
train_meta['Barratt_Barratt_P1_Edu'] = train_meta['Barratt_Barratt_P1_Edu'].astype('category')
train_meta['Barratt_Barratt_P1_Occ'] = train_meta['Barratt_Barratt_P1_Occ'].astype('category')
train_meta['Barratt_Barratt_P2_Edu'] = train_meta['Barratt_Barratt_P2_Edu'].astype('category')
train_meta['Barratt_Barratt_P2_Occ'] = train_meta['Barratt_Barratt_P2_Occ'].astype('category')
train_meta['participant_id'] = train_meta['participant_id'].astype('str')
train_meta['Basic_Demos_Enroll_Year'] = train_meta['Basic_Demos_Enroll_Year'].astype('category')

#handle missing data

from sklearn.impute import SimpleImputer

train_imp = train_meta.copy(deep=True)
test_imp  = test_meta.copy(deep=True)

num_cols = train_imp.select_dtypes(include=['int64', 'float64']).columns.tolist()
cat_cols = train_imp.select_dtypes(include=['category']).columns.tolist()
target_cols = ['ADHD_Outcome', 'Sex_F']
for target in target_cols:
    if target in num_cols:
        num_cols.remove(target)
median_vals = train_imp[num_cols].median()
train_imp[num_cols] = train_imp[num_cols].fillna(median_vals)
test_imp [num_cols] = test_imp [num_cols].fillna(median_vals)

for col in cat_cols:
  train_imp[col].fillna(train_imp[col].mode()[0], inplace=True)
  if col in test_imp:
    test_imp[col].fillna(test_imp[col].mode()[0], inplace=True)
train_imp[cat_cols] = pd.DataFrame(train_imp[cat_cols],
                                    columns=cat_cols,
                                   index=train_imp.index)

train_imp.to_csv('train_meta_imputed6.csv', index=False)

#process functional data
```

```
PCA_VAR=0.90
RNG = 42

edge_cols = [c for c in train_fun.columns if c != 'participant_id']
m         = int(np.sqrt(len(edge_cols)))          # square matrix dimension
tri_idx   = np.triu_indices(m, k=1)               # exclude diag & lower tri
keep_cols = [edge_cols[i*m + j] for i, j in zip(*tri_idx)]

train_ut  = train_fun[keep_cols]
test_ut   = test_fun [keep_cols]

#PCA fit on TRAIN
pca = PCA(n_components=PCA_VAR, random_state=RNG)
train_pcs = pca.fit_transform(train_ut)


pc_cols = [f'FC_PC{i}' for i in range(train_pcs.shape[1])]

train_fun_pcs = pd.DataFrame(train_pcs, columns=pc_cols).assign(
                   participant_id=train_fun['participant_id'])


#handling outliers using IQR method

def drop_outliers(df, numeric_columns):

    exclude = ['ADHD_Outcome','Sex_F']
    cols = [c for c in numeric_columns if c not in exclude]
    numeric_columns = [col for col in numeric_columns if col not in exclude]
    df_num = df[numeric_columns]
    Q1 = df_num.quantile(0.25)
    Q3 = df_num.quantile(0.75)
    IQR = Q3 - Q1

    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    mask = ~((df_num < lower) | (df_num > upper)).any(axis=1)
    return df.loc[mask].reset_index(drop=True)

train_no_outliers_nofun = drop_outliers(train_imp, num_cols)

#heatmap

df=train_no_outliers_nofun
#numeric_cols = df.select_dtypes(include=['number']).columns
```

```
# Compute correlation matrix
corr_matrix = df[num_cols].corr()

#Plot heatmap
plt.figure(figsize=(14, 12))
plt.imshow(corr_matrix, aspect='auto')
plt.colorbar()
plt.title('Feature Correlation Heatmap')
plt.xticks(range(len(num_cols)), num_cols, rotation=90)
plt.yticks(range(len(num_cols)), num_cols)
plt.tight_layout()
plt.show()


numeric_df = df.select_dtypes(include=[np.number])

# Compute absolute correlation matrix
corr = numeric_df.corr().abs()

# Mask the upper triangle
upper = corr.where(np.triu(np.ones(corr.shape), k=1).astype(bool))

# Find features with any correlation > 0.8
to_drop = [col for col in upper.columns if any(upper[col] > 0.8)]

print(f"Dropping {len(to_drop)} features:\n{to_drop}")

# Drop them
df_reduced = df.drop(columns=to_drop) #include functional

import seaborn as sns

# Settings
sns.set(style="whitegrid")
plt.figure(figsize=(6, 4))
sns.countplot(data=df_reduced, x='Sex_F', hue='ADHD_Outcome')
plt.title('ADHD Diagnosis by Sex')
plt.xlabel('Sex_F (0 = Male, 1 = Female)')
plt.ylabel('Count')
plt.legend(title='ADHD_Outcome', labels=['No ADHD (0)', 'ADHD (1)'])
plt.tight_layout()
plt.show()

sns.set(style="whitegrid")
plt.figure(figsize=(8, 5))
```

```
sns.boxplot(data=df_reduced, x='ADHD_Outcome', y='MRI_Track_Age_at_Scan')
plt.title('Age Distribution by ADHD Diagnosis')
plt.xlabel('ADHD_Outcome')
plt.ylabel('Age at Scan')
plt.tight_layout()
plt.show()

#merge functional

train_final = (df_reduced
               .merge(train_fun_pcs, on='participant_id'))

test_final  = (test_reduced
               .merge(test_fun_pcs,  on='participant_id'))

train_final.to_csv('train_final_full.csv', index=False)
test_final .to_csv('test_final_full.csv',  index=False)


#hot-code encode

train_enc = train_no_outliers_nofun.copy(deep=True)

train_dum = pd.get_dummies(train_enc[cat_cols], drop_first=True, dtype='uint8')

train_encoded_final = pd.concat(
        [train_enc.drop(columns=cat_cols), train_dum], axis=1)

train_encoded_final.to_csv('train_meta_encoded2.csv', index=False)

train_encoded_full = train_encoded_final.merge(train_fun_pcs, on='participant_id')
#est_encoded_full  = test_encoded_final.merge(test_fun_pcs,  on='participant_id')
train_encoded_full.to_csv('train_encoded_full.csv', index=False)

def evaluate_baseline(train_path):

    df = pd.read_csv(train_path)
    id_col = 'participant_id'
    target_cols = ['ADHD_Outcome', 'Sex_F']
    X = df.drop(columns=[id_col] + target_cols)

    skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

    # baseline models
    models = {
        'LogisticRegression': Pipeline([
            ('scaler', StandardScaler()),
            ('lr', LogisticRegression(max_iter=1000, solver='saga', n_jobs=-1))
        ]),
        'HistGradientBoosting': HistGradientBoostingClassifier(random_state=42)
    }

    results = []
    for target in target_cols:
        y = df[target]
        for name, model in models.items():
            scores = cross_val_score(model, X, y, cv=skf, scoring='roc_auc', n_jobs=-1)
            results.append({
                'target': target,
                'model': name,
                'mean_auc': scores.mean(),
                'std_auc': scores.std()
            })

    return pd.DataFrame(results)


df_results = evaluate_baseline('/content/drive/MyDrive/Datathon/Processed new/encoded/include functi
print(df_results)
#df_results.to_csv('baseline_results.csv', index=False)
#print("Baseline evaluation complete. Results saved to baseline_results.csv")

#xgboost muli model

from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import accuracy_score, roc_auc_score

from xgboost import XGBClassifier
from lightgbm import LGBMClassifier


df = pd.read_csv('/content/drive/MyDrive/Datathon/Processed new2/train_encoded_full (2).csv')


X = df.drop(['participant_id', 'ADHD_Outcome', 'Sex_F'], axis=1)
y = df[['ADHD_Outcome', 'Sex_F']]


X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.20,
    random_state=42,
    stratify=list(zip(y['ADHD_Outcome'], y['Sex_F']))
)


multi_xgb = MultiOutputClassifier(
    XGBClassifier(
        use_label_encoder=False,
        eval_metric='logloss',
        random_state=42
    )
)
multi_xgb.fit(X_train, y_train)


multi_lgb = MultiOutputClassifier(
    LGBMClassifier(random_state=42)
)
multi_lgb.fit(X_train, y_train)

def evaluate_model(multi_clf, X_test, y_test, name):
    y_pred = multi_clf.predict(X_test)
    y_proba = multi_clf.predict_proba(X_test)

    for idx, col in enumerate(y_test.columns):
        acc = accuracy_score(y_test[col], y_pred[:, idx])
        # For ROC AUC, need the positive-class probability array
        auc = roc_auc_score(y_test[col], y_proba[idx][:,1])
        print(f"{name} | {col}:")
        print(f"    Accuracy = {acc:.4f}")
        print(f"    ROC AUC  = {auc:.4f}")
    print("-" * 40)

print("\n=== XGBoost Multi-Output ===")
evaluate_model(multi_xgb, X_test, y_test, "XGBoost")

print("\n=== LightGBM Multi-Output ===")
evaluate_model(multi_lgb, X_test, y_test, "LightGBM")

#seperate

#baseline
import pandas as pd
import numpy as np
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import f1_score, classification_report
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier


df   = pd.read_csv('/content/drive/MyDrive/Datathon/Processed new/encoded/no functional/train_meta_e

ID_COL    = 'participant_id'
TARGET_A  = 'ADHD_Outcome'
TARGET_S  = 'Sex_F'

X = df.drop(columns=[ID_COL, TARGET_A, TARGET_S])
y_ad = df[TARGET_A]
y_se = df[TARGET_S]

X_tr, X_val, y_ad_tr, y_ad_val, y_se_tr, y_se_val = train_test_split(
    X, y_ad, y_se,
    test_size=0.20,
    stratify=y_ad,
    random_state=42
)

cat_cols = X_tr.select_dtypes(include=['object','category']).columns.tolist()
if cat_cols:
    oe = OrdinalEncoder()
    X_tr[cat_cols]  = oe.fit_transform(X_tr[cat_cols])
    X_val[cat_cols] = oe.transform(X_val[cat_cols])

xgb = XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
xgb.fit(X_tr, y_ad_tr)
pred_ad_val = xgb.predict(X_val)
f1_ad = f1_score(y_ad_val, pred_ad_val, average='macro')

print(f"\nADHD_Outcome hold-out macro-F1: {f1_ad:.3f}")
print(classification_report(y_ad_val, pred_ad_val,
      target_names=['Control','ADHD']))
```

```
lgbm = LGBMClassifier(
    objective='binary',
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
lgbm.fit(X_tr, y_se_tr)
pred_se_val = lgbm.predict(X_val)
f1_se = f1_score(y_se_val, pred_se_val, average='macro')

print(f"\nSex_F hold-out macro-F1: {f1_se:.3f}")
print(classification_report(y_se_val, pred_se_val,
      target_names=['Male','Female']))

import pandas as pd
import numpy as np
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import f1_score, classification_report
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

# 0) Load only the processed train file
df   = pd.read_csv('/content/drive/MyDrive/Datathon/Processed new/encoded/include functional/train_e


ID_COL    = 'participant_id'
TARGET_A  = 'ADHD_Outcome'
TARGET_S  = 'Sex_F'

X = df.drop(columns=[ID_COL, TARGET_A, TARGET_S])
y_ad = df[TARGET_A]
y_se = df[TARGET_S]

X_tr, X_val, y_ad_tr, y_ad_val, y_se_tr, y_se_val = train_test_split(
    X, y_ad, y_se,
    test_size=0.20,
    stratify=y_ad,
    random_state=42
)

cat_cols = X_tr.select_dtypes(include=['object','category']).columns.tolist()
```

```
if cat_cols:
    oe = OrdinalEncoder()
    X_tr[cat_cols]  = oe.fit_transform(X_tr[cat_cols])
    X_val[cat_cols] = oe.transform(X_val[cat_cols])

xgb = XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
xgb.fit(X_tr, y_ad_tr)
pred_ad_val = xgb.predict(X_val)
f1_ad = f1_score(y_ad_val, pred_ad_val, average='macro')

print(f"\nADHD_Outcome hold-out macro-F1: {f1_ad:.3f}")
print(classification_report(y_ad_val, pred_ad_val,
      target_names=['Control','ADHD']))

lgbm = LGBMClassifier(
    objective='binary',
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
lgbm.fit(X_tr, y_se_tr)
pred_se_val = lgbm.predict(X_val)
f1_se = f1_score(y_se_val, pred_se_val, average='macro')

print(f"\nSex_F hold-out macro-F1: {f1_se:.3f}")
print(classification_report(y_se_val, pred_se_val,
      target_names=['Male','Female']))

import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
```

```
RNG       = 42

df = pd.read_csv('/content/drive/MyDrive/Datathon/Processed new/encoded/no functional/train_meta_enc

feature_cols = [c for c in df.columns
                if c not in ('participant_id','ADHD_Outcome','Sex_F')]
X = df[feature_cols]
y_adhd = df['ADHD_Outcome']
y_sex  = df['Sex_F']

X = X[mask]
y_adhd = y_adhd[mask]
y_sex  = y_sex[mask]

X = X.fillna(0)

corr_adhd = X.corrwith(y_adhd).abs().sort_values(ascending=False)
corr_sex  = X.corrwith(y_sex).abs().sort_values(ascending=False)

print("Top 5 |corr| with ADHD:")
print(corr_adhd.head(5), "\n")
print("Top 5 |corr| with Sex_F:")
print(corr_sex.head(5), "\n")

plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
corr_adhd.head(10).plot.barh(title='|corr| vs ADHD')
plt.gca().invert_yaxis()
plt.subplot(1,2,2)
corr_sex.head(10).plot.barh(title='|corr| vs Sex_F')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()

def plot_importances(X, y, title):
    rf = RandomForestClassifier(n_estimators=200, random_state=RNG)
    rf.fit(X, y)
    imp = pd.Series(rf.feature_importances_, index=X.columns).sort_values(ascending=False)
    print(f"Top 5 importances for {title}:")
    print(imp.head(5), "\n")
    plt.figure(figsize=(6,4))
    imp.head(10).plot.barh(title=f'RF importances: {title}')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
```

```
plot_importances(X, y_adhd, 'ADHD_Outcome')
plot_importances(X, y_sex,    'Sex_F')

import pandas as pd
import numpy as np
from pathlib import Path
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier


META_FILE   = '/content/drive/MyDrive/Datathon/Processed new/encoded/no functional/train_meta_encode
FC_FILE     = '/content/drive/MyDrive/Datathon/TRAIN_NEW/TRAIN_FUNCTIONAL_CONNECTOME_MATRICES_new_36
PCA_VAR     = 0.90    # keep 90% variance
SPARSITY_TH = 0.2     # threshold for \weak" edges
RNG         = 42

meta = pd.read_csv(META_FILE)

fc   = pd.read_csv(FC_FILE)

edge_cols = [c for c in fc.columns if c!='participant_id']
m         = int(np.sqrt(len(edge_cols)))
tri_idx   = np.triu_indices(m, k=1)
pairs     = list(zip(*tri_idx))
keep_cols = [edge_cols[i*m + j] for i,j in pairs]
ut        = fc[keep_cols]

pca       = PCA(n_components=PCA_VAR, random_state=RNG)
pcs       = pca.fit_transform(ut)
pc_cols   = [f'FC_PC{i}' for i in range(pcs.shape[1])]
pcs_df    = pd.DataFrame(pcs, columns=pc_cols).assign(participant_id=fc['participant_id'])

# Graph-summary features
global_strength = ut.mean(axis=1).rename('global_strength')
sparsity        = (ut.abs() < SPARSITY_TH).mean(axis=1).rename('sparsity')

# Define regions by ROI indices
region_nodes = {
    'frontal':   list(range(0,10)),
    'parietal':  list(range(10,18)),
    'occipital': list(range(18,26)),
    'temporal':  list(range(26,36))
}

def region_means(ut_df, pairs, region_nodes):
    out = {}
    for region, nodes in region_nodes.items():
        idxs = [k for k,(i,j) in enumerate(pairs) if i in nodes and j in nodes]
        out[f'{region}_mean'] = ut_df.iloc[:,idxs].mean(axis=1)
    return pd.DataFrame(out, index=ut_df.index)

region_df = region_means(ut, pairs, region_nodes)

# Merge into one table by participant_id
global_df   = pd.DataFrame({
    'participant_id': fc['participant_id'],
    'global_strength': global_strength
})
sparsity_df = pd.DataFrame({
    'participant_id': fc['participant_id'],
    'sparsity':      train_sparsity
})
region_df   = region_df.reset_index(drop=True).assign(
    participant_id = fc['participant_id']
)

df = (
    meta
      .merge(pcs_df,      on='participant_id', how='left')
      .merge(global_df,   on='participant_id', how='left')
      .merge(sparsity_df, on='participant_id', how='left')
      .merge(region_df,   on='participant_id', how='left')
)

region_cols  = list(region_df.columns)       # ['frontal_mean', 'parietal_mean', ...]
region_cols = [c for c in region_cols if c.endswith('_mean')]
summary_feats = ['global_strength','sparsity'] + region_cols
print(region_cols)
X = df[summary_feats].copy()

y_adhd = df['ADHD_Outcome'].astype(int)
y_sex  = df['Sex_F'].astype(int)

corr_adhd = X.corrwith(y_adhd).abs().sort_values(ascending=False)
corr_sex  = X.corrwith(y_sex ).abs().sort_values(ascending=False)

print("Top 5 |corr| with ADHD:\n", corr_adhd.head(5), "\n")
print("Top 5 |corr| with Sex_F:\n", corr_sex.head(5), "\n")

from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
```

```
def plot_importances(X, y, title):
    rf = RandomForestClassifier(n_estimators=200, random_state=RNG)
    rf.fit(X, y)
    imp = pd.Series(rf.feature_importances_, index=X.columns).sort_values(ascending=False)
    print(f"Top 5 importances for {title}:\n{imp.head(5)}\n")
    plt.figure(figsize=(6,4))
    imp.head(10).plot.barh(title=f'RF importances: {title}')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()

plot_importances(X, y_adhd, 'ADHD_Outcome')
plot_importances(X, y_sex,    'Sex_F')

import xgboost as xgb
from lightgbm import LGBMClassifier

#  1) Assume `df` already has these six summary columns + labels
# summary columns: global_strength, sparsity, frontal_mean, parietal_mean, occipital_mean, temporal_
# labels: ADHD_Outcome, Sex_F

# 2) Define the two feature subsets
adhd_feats = ['sparsity', 'parietal_mean', 'temporal_mean', 'occipital_mean', 'frontal_mean']
sex_feats  = ['global_strength', 'parietal_mean', 'frontal_mean', 'sparsity', 'temporal_mean']

# 3) Prepare X/y for ADHD and Sex
X_adhd = df[adhd_feats].fillna(0)
y_adhd = df['ADHD_Outcome'].astype(int)

X_sex  = df[sex_feats].fillna(0)
y_sex  = df['Sex_F'].astype(int)

# 4) Train/test split for ADHD
X_tr_a, X_te_a, y_tr_a, y_te_a = train_test_split(
    X_adhd, y_adhd, test_size=0.2,
    stratify=y_adhd, random_state=42
)

# 5) XGBoost for ADHD_Outcome
xgb_clf = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    learning_rate=0.05,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
xgb_clf.fit(X_tr_a, y_tr_a)
pred_a = xgb_clf.predict(X_te_a)

print("=== ADHD_Outcome (XGBoost) ===")
print(classification_report(y_te_a, pred_a, target_names=['Control','ADHD']))
print("Macro-F1 ADHD:", f1_score(y_te_a, pred_a, average='macro'))

# 6) Train/test split for Sex
X_tr_s, X_te_s, y_tr_s, y_te_s = train_test_split(
    X_sex, y_sex, test_size=0.2,
    stratify=y_sex, random_state=42
)

# 7) LightGBM for Sex_F
lgbm_clf = LGBMClassifier(
    objective='binary',
    learning_rate=0.05,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    class_weight='balanced',
    random_state=42
)
lgbm_clf.fit(X_tr_s, y_tr_s)
pred_s = lgbm_clf.predict(X_te_s)

print("\n=== Sex_F (LightGBM) ===")
print(classification_report(y_te_s, pred_s, target_names=['Male','Female']))
print("Macro-F1 Sex_F:", f1_score(y_te_s, pred_s, average='macro'))


#baseline
import pandas as pd
import numpy as np
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import f1_score, classification_report
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

# 0) Load only the processed train file
```

```
df   = pd.read_csv('/content/drive/MyDrive/Datathon/Processed new/encoded/no functional/train_meta_e

# 1) Split out features & labels
ID_COL    = 'participant_id'
TARGET_A  = 'ADHD_Outcome'
TARGET_S  = 'Sex_F'

X = df.drop(columns=[ID_COL, TARGET_A, TARGET_S])
y_ad = df[TARGET_A]
y_se = df[TARGET_S]

# 2) 80/20 train-validation split (stratified on ADHD_Outcome)
X_tr, X_val, y_ad_tr, y_ad_val, y_se_tr, y_se_val = train_test_split(
    X, y_ad, y_se,
    test_size=0.20,
    stratify=y_ad,
    random_state=42
)

# 3) Ordinal-encode any categorical columns
cat_cols = X_tr.select_dtypes(include=['object','category']).columns.tolist()
if cat_cols:
    oe = OrdinalEncoder()
    X_tr[cat_cols]  = oe.fit_transform(X_tr[cat_cols])
    X_val[cat_cols] = oe.transform(X_val[cat_cols])

# 4) Train XGBoost on ADHD_Outcome (no early stopping)
xgb = XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
xgb.fit(X_tr, y_ad_tr)
pred_ad_val = xgb.predict(X_val)
f1_ad = f1_score(y_ad_val, pred_ad_val, average='macro')

print(f"\nADHD_Outcome hold-out macro-F1: {f1_ad:.3f}")
print(classification_report(y_ad_val, pred_ad_val,
      target_names=['Control','ADHD']))

# 5) Train LightGBM on Sex_F (no early stopping)
```

```
lgbm = LGBMClassifier(
    objective='binary',
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
lgbm.fit(X_tr, y_se_tr)
pred_se_val = lgbm.predict(X_val)
f1_se = f1_score(y_se_val, pred_se_val, average='macro')

print(f"\nSex_F hold-out macro-F1: {f1_se:.3f}")
print(classification_report(y_se_val, pred_se_val,
      target_names=['Male','Female']))


import pandas as pd
import numpy as np
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import f1_score, classification_report
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

# 0) Load only the processed train file
df   = pd.read_csv('/content/drive/MyDrive/Datathon/Processed new/encoded/include functional/train_e

# 1) Split out features & labels
ID_COL    = 'participant_id'
TARGET_A  = 'ADHD_Outcome'
TARGET_S  = 'Sex_F'

X = df.drop(columns=[ID_COL, TARGET_A, TARGET_S])
y_ad = df[TARGET_A]
y_se = df[TARGET_S]

# 2) 80/20 train-validation split (stratified on ADHD_Outcome)
X_tr, X_val, y_ad_tr, y_ad_val, y_se_tr, y_se_val = train_test_split(
    X, y_ad, y_se,
    test_size=0.20,
    stratify=y_ad,
    random_state=42
)
```

```
# 3) Ordinal-encode any categorical columns
cat_cols = X_tr.select_dtypes(include=['object','category']).columns.tolist()
if cat_cols:
    oe = OrdinalEncoder()
    X_tr[cat_cols]  = oe.fit_transform(X_tr[cat_cols])
    X_val[cat_cols] = oe.transform(X_val[cat_cols])

# 4) Train XGBoost on ADHD_Outcome (no early stopping)
xgb = XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
xgb.fit(X_tr, y_ad_tr)
pred_ad_val = xgb.predict(X_val)
f1_ad = f1_score(y_ad_val, pred_ad_val, average='macro')

print(f"\nADHD_Outcome hold-out macro-F1: {f1_ad:.3f}")
print(classification_report(y_ad_val, pred_ad_val,
      target_names=['Control','ADHD']))

# 5) Train LightGBM on Sex_F (no early stopping)
lgbm = LGBMClassifier(
    objective='binary',
    learning_rate=0.05,
    n_estimators=500,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
lgbm.fit(X_tr, y_se_tr)
pred_se_val = lgbm.predict(X_val)
f1_se = f1_score(y_se_val, pred_se_val, average='macro')

print(f"\nSex_F hold-out macro-F1: {f1_se:.3f}")
print(classification_report(y_se_val, pred_se_val,
      target_names=['Male','Female']))


# Logistic Regression
```

```
- Meta-only

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import matplotlib.pyplot as plt

train_meta = pd.read_csv('/content/drive/MyDrive/datathon/Datathon/Processed new/encoded/no functior
#test_meta  = pd.read_csv('/Users/chenliu/Desktop/project/Processed data/encoded/functional not in/t
print(train_meta.columns.tolist())
#print(test_meta.columns.tolist())

# split labels and features
y_adhd = train_meta['ADHD_Outcome']
y_sex  = train_meta['Sex_F']
X = train_meta.drop(columns=['participant_id', 'ADHD_Outcome', 'Sex_F'])

print("X Shape:", X.shape)
print("ADHD Distribution:\n", y_adhd.value_counts())
print("Sex Distribution:\n", y_sex .value_counts())


# split dataset for test and training
from sklearn.model_selection import train_test_split
X_tr, X_te, y_sex_tr, y_sex_te, y_adhd_tr, y_adhd_te = train_test_split(
    X, y_sex, y_adhd,
    test_size=0.2,
    stratify=y_sex,
    random_state=42
)

print("Number of training samples:", X_tr.shape[0])
print("Number of test samples:", X_te.shape[0])
print("Training samples ADHD Distribution:\n", y_adhd_tr.value_counts())
print("Test samples ADHD Distribution:\n", y_adhd_te.value_counts())


# Standardize
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_tr_scaled = scaler.fit_transform(X_tr)
X_te_scaled = scaler.transform(X_te)


# Training model
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

# Training ADHD Model
clf_adhd = LogisticRegression(max_iter=1000)
clf_adhd.fit(X_tr_scaled, y_adhd_tr)

# Test ADHD Model
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
    roc_curve,
    auc)

# Compute ROC
clf_sex  = LogisticRegression(max_iter=1000, class_weight='balanced')
clf_adhd = LogisticRegression(max_iter=1000, class_weight='balanced')

clf_sex .fit(X_tr_scaled, y_sex_tr)
clf_adhd.fit(X_tr_scaled, y_adhd_tr)

p_sex  = clf_sex .predict_proba(X_te_scaled)[:,1]
p_adhd = clf_adhd.predict_proba(X_te_scaled)[:,1]

print("Meta-only → Sex  AUC:", roc_auc_score(y_sex_te,  p_sex ))
print("Meta-only → ADHD AUC:", roc_auc_score(y_adhd_te, p_adhd))

# Threshold Metrics ADHD
threshold = 0.5
y_adhd_pred = (p_adhd >= threshold).astype(int)
acc_a   = accuracy_score(y_adhd_te, y_adhd_pred)
prec_a  = precision_score(y_adhd_te, y_adhd_pred)
rec_a   = recall_score(y_adhd_te, y_adhd_pred)
f1_a    = f1_score(y_adhd_te, y_adhd_pred)
cm_a    = confusion_matrix(y_adhd_te, y_adhd_pred)

print("=== Threshold-based Metrics (threshold=0.5) ===")
print(f"Accuracy : {acc_a:.3f}")
print(f"Precision: {prec_a:.3f}")
print(f"Recall   : {rec_a:.3f}")
print(f"F1-score : {f1_a:.3f}")
print("Confusion Matrix:")
print(cm_a)
```

```
# Threshold-based Metrics Sex
y_sex_pred = (p_sex >= 0.5).astype(int)
acc_s  = accuracy_score(y_sex.loc[X_te.index], y_sex_pred)
prec_s = precision_score(y_sex.loc[X_te.index], y_sex_pred)
rec_s  = recall_score(y_sex.loc[X_te.index], y_sex_pred)
f1_s   = f1_score(y_sex.loc[X_te.index], y_sex_pred)
cm_s   = confusion_matrix(y_sex.loc[X_te.index], y_sex_pred)

print("\n=== Sex Threshold Metrics (0.5) ===")
print(f"Accuracy : {acc_s :.3f}")
print(f"Precision: {prec_s:.3f}")
print(f"Recall   : {rec_s :.3f}")
print(f"F1-score : {f1_s :.3f}")
print("Confusion Matrix:\n", cm_s)

# Pic 1

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# ROC curves for ADHD & Sex
fpr_a, tpr_a, _ = roc_curve(y_adhd_te, p_adhd)
roc_auc_a   = auc(fpr_a, tpr_a)

fpr_s, tpr_s, _ = roc_curve(y_sex_te, p_sex)
roc_auc_s   = auc(fpr_s, tpr_s)

plt.figure(figsize=(6,6))
plt.plot(fpr_a, tpr_a, label=f'ADHD ROC (AUC = {roc_auc_a:.2f})')
plt.plot(fpr_s, tpr_s, label=f'Sex  ROC (AUC = {roc_auc_s:.2f})')
plt.plot([0,1],[0,1],'--', label='Chance')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(' Logistic Model Baseline ROC Curves')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

# Bar chart of threshold-based metrics
labels = ['ADHD', 'Sex']
metrics = {
    'Accuracy' : [acc_a, acc_s],
    'Precision': [prec_a, prec_s],
    'Recall'   : [rec_a, rec_s],
    'F1-score' : [f1_a, f1_s],
}

x     = np.arange(len(labels))
width = 0.2

plt.figure(figsize=(8,5))
for i, (name, vals) in enumerate(metrics.items()):
    plt.bar(x + i*width, vals, width, label=name)

plt.xticks(x + 1.5*width, labels)
plt.ylim(0,1)
plt.ylabel('Score')
plt.title('Threshold-based Metrics (threshold=0.5)')
plt.legend(loc='upper center', ncol=4, bbox_to_anchor=(0.5, 1.12))
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Confusion matrix heatmaps
fig, axes = plt.subplots(1,2, figsize=(20,4))

for ax, cm, title in zip(axes, [cm_a, cm_s], ['ADHD CM','Sex CM']):
    im = ax.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    ax.set_title(title)
    ax.set_xlabel('Predicted label')
    ax.set_ylabel('True label')
    # annotate counts
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, cm[i, j], ha='center', va='center')
fig.colorbar(im, ax=axes.ravel().tolist(), shrink=0.6)
plt.tight_layout()
plt.show()


- Meta + FC

train_df = pd.read_csv('/content/drive/MyDrive/datathon/Datathon/Processed new/encoded/include funct
#test_df  = pd.read_csv('/Users/chenliu/Desktop/untitled folder/Processed data/encoded/funtional in/
print(train_df.columns.tolist())
#print(test_df.columns.tolist())

import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing  import StandardScaler
from sklearn.linear_model   import LogisticRegression
from sklearn.metrics        import roc_auc_score

# split labels and features
X = train_df.drop(columns=['participant_id','Sex_F','ADHD_Outcome'])
y_sex  = train_df['Sex_F']
y_adhd = train_df['ADHD_Outcome']

# # split training and test dataset
X_tr, X_te, y_sex_tr, y_sex_te, y_adhd_tr, y_adhd_te = train_test_split(
    X, y_sex, y_adhd,
    test_size=0.2,
    stratify=y_sex,
    random_state=42
)

# Standardize
scaler = StandardScaler()
X_tr_scaled = scaler.fit_transform(X_tr)
X_te_scaled = scaler.transform(X_te)

# Training and Test Model
clf_sex  = LogisticRegression(max_iter=1000, class_weight='balanced')
clf_adhd = LogisticRegression(max_iter=1000, class_weight='balanced')

clf_sex .fit(X_tr_scaled, y_sex_tr)
clf_adhd.fit(X_tr_scaled, y_adhd_tr)

p_sex  = clf_sex .predict_proba(X_te_scaled)[:,1]
p_adhd = clf_adhd.predict_proba(X_te_scaled)[:,1]

print("Meta+FC → Sex  AUC:", roc_auc_score(y_sex_te,  p_sex ))
print("Meta+FC → ADHD AUC:", roc_auc_score(y_adhd_te, p_adhd))


# pic 2

# ROC curves for Sex & ADHD
fpr_sex, tpr_sex, _ = roc_curve(y_sex_te, p_sex)
roc_auc_sex = auc(fpr_sex, tpr_sex)

fpr_adhd, tpr_adhd, _ = roc_curve(y_adhd_te, p_adhd)
roc_auc_adhd = auc(fpr_adhd, tpr_adhd)
```

```
plt.figure(figsize=(6,6))
plt.plot(fpr_sex, tpr_sex, label=f'Sex ROC (AUC = {roc_auc_sex:.2f})')
plt.plot(fpr_adhd, tpr_adhd, label=f'ADHD ROC (AUC = {roc_auc_adhd:.2f})')
plt.plot([0,1], [0,1], '--', label='Chance')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(' Logistic Model Meta+FC ROC Curves')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

# Threshold-based bar chart (threshold=0.5)
threshold = 0.5
y_sex_pred = (p_sex >= threshold).astype(int)
y_adhd_pred = (p_adhd >= threshold).astype(int)

metrics = {
    'Sex': {
        'Accuracy':  accuracy_score(y_sex_te, y_sex_pred),
        'Precision': precision_score(y_sex_te, y_sex_pred),
        'Recall':    recall_score(y_sex_te, y_sex_pred),
        'F1-score':  f1_score(y_sex_te, y_sex_pred),
    },
    'ADHD': {
        'Accuracy':  accuracy_score(y_adhd_te, y_adhd_pred),
        'Precision': precision_score(y_adhd_te, y_adhd_pred),
        'Recall':    recall_score(y_adhd_te, y_adhd_pred),
        'F1-score':  f1_score(y_adhd_te, y_adhd_pred),
    },
}

labels = list(metrics.keys())
scores = {m: [metrics[label][m] for label in labels] for m in metrics['Sex'].keys()}

x = np.arange(len(labels))
width = 0.2

plt.figure(figsize=(8,5))
for i, (name, vals) in enumerate(scores.items()):
    plt.bar(x + i*width - 1.5*width, vals, width, label=name)

plt.xticks(x, labels)
plt.ylim(0,1)
plt.ylabel('Score')
plt.title('Meta+FC Threshold-based Metrics (threshold=0.5)')
plt.legend(loc='upper center', ncol=4, bbox_to_anchor=(0.5, 1.12))
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Confusion matrices
fig, axes = plt.subplots(1, 2, figsize=(20,4))
cms = [
    confusion_matrix(y_sex_te, y_sex_pred),
    confusion_matrix(y_adhd_te, y_adhd_pred)
]
titles = ['Sex Confusion Matrix', 'ADHD Confusion Matrix']

for ax, cm, title in zip(axes, cms, titles):
    im = ax.imshow(cm, interpolation='nearest')
    ax.set_title(title)
    ax.set_xlabel('Predicted label')
    ax.set_ylabel('True label')
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, cm[i, j], ha='center', va='center')
fig.colorbar(im, ax=axes.ravel().tolist(), shrink=0.6)
plt.tight_layout()
plt.show()


## Results Comparison

| Pipeline      | Sex AUC | ADHD AUC |
|---------------|--------:|---------:|
| **Meta-only** |  0.570  |   0.874  |
| **Meta + FC** |  0.615  |   0.668  |

### Key Takeaways

1. **Sex Prediction**
   - Adding functional connectivity features improved Sex AUC from **0.570 → 0.615**, indicating FC

2. **ADHD Prediction**
   - ADHD AUC dropped from **0.874 → 0.668** when FC was added, suggesting the raw FC matrix introdu

- Feature Engineering for Meta-only Data

import pandas as pd
from sklearn.model_selection   import train_test_split
from sklearn.preprocessing    import StandardScaler, KBinsDiscretizer, PolynomialFeatures
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.linear_model     import LogisticRegression
from sklearn.metrics          import roc_auc_score

df = train_meta

# Public feature engineering
X0 = df.drop(columns=['participant_id','Sex_F','ADHD_Outcome']).copy()

# EHQ binning
kb = KBinsDiscretizer(n_bins=3, encode='onehot-dense', strategy='quantile')
ehq_bins = kb.fit_transform(X0[['EHQ_EHQ_Total']])
X0[['EHQ_low','EHQ_mid','EHQ_high']] = ehq_bins

# APQ sum and interaction
apq_cols = [c for c in X0.columns if c.startswith('APQ_P_APQ_P_')]
X0['Sum_APQ']    = X0[apq_cols].sum(axis=1)
X0['CVxAPQ_sum'] = X0['ColorVision_CV_Score'] * X0['Sum_APQ']

# Polynomial expansion
poly = PolynomialFeatures(degree=2, interaction_only=False, include_bias=False)
poly_in = X0[['EHQ_EHQ_Total','ColorVision_CV_Score','Sum_APQ']]
poly_ft = poly.fit_transform(poly_in)
poly_nm = poly.get_feature_names_out(poly_in.columns)
X0[poly_nm] = poly_ft

# Multi-task independent pipelines for ADHD and Sex
results = {}
roc_data = {}
for task, y in (("ADHD", df['ADHD_Outcome']), ("Sex", df['Sex_F'])):
    # Feature selection per task
    selector = SelectKBest(f_classif, k=20)
    X_task  = selector.fit_transform(X0, y)
    cols_sel = X0.columns[selector.get_support()]

    # Split into train/test stratified by current task label
    X_tr, X_te, y_tr, y_te = train_test_split(
        X_task, y,
        test_size=0.2,
        stratify=y,
        random_state=42
    )

    # Standardize
    scaler = StandardScaler()
    X_tr_s = scaler.fit_transform(X_tr)
    X_te_s = scaler.transform(X_te)

    # Traning and Test Model
    clf = LogisticRegression(max_iter=1000, class_weight='balanced')
    clf.fit(X_tr_s, y_tr)
    p = clf.predict_proba(X_te_s)[:,1]
    auc = roc_auc_score(y_te, p)

    results[task] = auc
    fpr, tpr, _ = roc_curve(y_te, p)
    roc_data[task]  = (fpr, tpr)

print("Multi-task Results (separate stratify & selection):")
for task, auc in results.items():
    print(f"  {task:4s} AUC: {auc:.3f}")


# pic 3

# 3) Plot ROC curves
plt.figure()
for task, (fpr, tpr) in roc_data.items():
    plt.plot(fpr, tpr, label=f"{task} (AUC={results[task]:.3f})")
plt.plot([0,1], [0,1], linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistic Model FE ROC Curves")
plt.legend()
plt.show()

# 4) Plot AUC comparison bar chart
plt.figure()
tasks = list(results.keys())
aucs  = [results[t] for t in tasks]
plt.bar(tasks, aucs)
plt.ylim(0, 1)
plt.xlabel("Task")
plt.ylabel("AUC")
plt.title("AUC Comparison")
for i, v in enumerate(aucs):
    plt.text(i, v + 0.02, f"{v:.3f}", ha='center')
plt.show()

## Logistic Test Comparison

| Pipeline                                  | Sex AUC | ADHD AUC |
|-----------------------------------------|--------:|---------:|
| **Meta-only (baseline)**                |   0.570 |    0.874 |
| **Meta + FC (raw concatenation)**       |   0.615 |    0.668 |
| **Multi-task w/ separate selection**    |   0.658 |    0.870 |



#  MLP


- Baseline: Encoded + Functional Not In

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import roc_auc_score

df = pd.read_csv('/content/drive/MyDrive/datathon/Datathon/Processed new/encoded/no functional/trai
X = df.drop(columns=['participant_id','ADHD_Outcome','Sex_F'])
y_sex = df['Sex_F']
y_adhd = df['ADHD_Outcome']

# 80/20 split dataset
X_tr, X_hold, y_sex_tr, y_sex_hold, y_adhd_tr, y_adhd_hold = train_test_split(
    X, y_sex, y_adhd,
    test_size=0.2,
    stratify=y_sex,
    random_state=42
)

# Standardize
scaler = StandardScaler()
X_tr_scaled   = scaler.fit_transform(X_tr)
X_hold_scaled = scaler.transform(X_hold)

# build a Baseline MLP
mlp_base = MLPClassifier(
    hidden_layer_sizes=(100,),
    max_iter=500,
    random_state=42
)
```

```
# Fit Sex Model
cv_scores_sex = cross_val_score(
    mlp_base,
    X_tr_scaled,
    y_sex_tr,
    cv=5,
    scoring='roc_auc',
    n_jobs=-1
)
print(f"Baseline MLP Sex 5-fold CV AUC: {cv_scores_sex.mean():.3f} ± {cv_scores_sex.std():.3f}")

mlp_base.fit(X_tr_scaled, y_sex_tr)
p_sex_hold = mlp_base.predict_proba(X_hold_scaled)[:,1]
hold_auc_sex = roc_auc_score(y_sex_hold, p_sex_hold)
print(f"Baseline MLP Sex Hold-out AUC: {hold_auc_sex:.3f}")

# Fit ADHD Model
cv_scores_adhd = cross_val_score(
    mlp_base,
    X_tr_scaled,
    y_adhd_tr,
    cv=5,
    scoring='roc_auc',
    n_jobs=-1
)
print(f"Baseline MLP ADHD 5-fold CV AUC: {cv_scores_adhd.mean():.3f} ± {cv_scores_adhd.std():.3f}")

mlp_base.fit(X_tr_scaled, y_adhd_tr)
p_adhd_hold = mlp_base.predict_proba(X_hold_scaled)[:,1]
hold_auc_adhd = roc_auc_score(y_adhd_hold, p_adhd_hold)
print(f"Baseline MLP ADHD Hold-out AUC: {hold_auc_adhd:.3f}")


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
import numpy as np

metrics = {}
for task, y_true, y_proba in [('ADHD', y_adhd_hold, p_adhd_hold),
                              ('Sex',  y_sex_hold,  p_sex_hold)]:
    y_pred = (y_proba >= 0.5).astype(int)
    metrics[task] = {
        'Accuracy':  accuracy_score(y_true, y_pred),
        'Precision': precision_score(y_true, y_pred),
        'Recall':    recall_score(y_true, y_pred),
        'F1-score':  f1_score(y_true, y_pred)
    }

# plot bar chat
labels = ['Accuracy', 'Precision', 'Recall', 'F1-score']
x = np.arange(len(labels))
width = 0.35

fig, ax = plt.subplots(figsize=(8,5))
adhd_vals = [metrics['ADHD'][m] for m in labels]
sex_vals  = [metrics['Sex'][m]  for m in labels]

ax.bar(x - width/2, adhd_vals, width, label='ADHD')
ax.bar(x + width/2, sex_vals,  width, label='Sex')

ax.set_ylabel('Score')
ax.set_title(' Baseline Threshold-based Metrics (threshold=0.5)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.set_ylim(0, 1)
ax.legend()
ax.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# plot ROC curve
fpr_adhd, tpr_adhd, _ = roc_curve(y_adhd_hold, p_adhd_hold)
roc_auc_adhd = auc(fpr_adhd, tpr_adhd)
fpr_sex, tpr_sex, _   = roc_curve(y_sex_hold, p_sex_hold)
roc_auc_sex  = auc(fpr_sex, tpr_sex)

plt.figure(figsize=(6,6))
plt.plot(fpr_adhd, tpr_adhd, label=f'ADHD ROC (AUC = {roc_auc_adhd:.2f})', lw=2)
plt.plot(fpr_sex,  tpr_sex,  label=f'Sex ROC (AUC = {roc_auc_sex:.2f})', lw=2)
plt.plot([0,1], [0,1], linestyle=':', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('MLP Baseline ROC Curves')
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

- Not Encoded + Functional In PCA

```
import numpy as np
from sklearn.decomposition import PCA
file_path = '/content/drive/MyDrive/datathon/Datathon/Processed new/encoded/include functional/trai
df = pd.read_csv(file_path)

# Separate Meta and FC Features
label_cols = ['participant_id', 'ADHD_Outcome', 'Sex_F']
prefixes = [
    'EHQ_', 'ColorVision_', 'APQ_P_APQ_P_', 'SDQ_SDQ_',
    'Barratt_Barratt_', 'Basic_Demos_', 'PreInt_Demos_', 'MRI_Track_']
all_cols = df.columns.tolist()
meta_cols = [
    c for c in all_cols
    if any(c.startswith(pref) for pref in prefixes)
    and c not in label_cols]
fc_cols = [c for c in all_cols if c.startswith('FC_')]

# get label
X_meta = df[meta_cols].values
X_fc   = df[fc_cols].values
y_sex  = df['Sex_F']
y_adhd = df['ADHD_Outcome']

# Standardize
X_meta_s = StandardScaler().fit_transform(X_meta)
X_fc_s   = StandardScaler().fit_transform(X_fc)

# PCA on FC → 50
X_fc_pca = PCA(n_components=50, random_state=42).fit_transform(X_fc_s)
X_all = np.hstack([X_meta_s, X_fc_pca])

# 7) 80/20 split datasets
X_tr, X_val, y_sex_tr, y_sex_val, y_adhd_tr, y_adhd_val = train_test_split(
    X_all, y_sex, y_adhd,
    test_size=0.2, stratify=y_sex, random_state=42
)

# Training Model
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)

# Sex Model
cv_sex   = cross_val_score(mlp, X_tr, y_sex_tr, cv=5, scoring='roc_auc')
```

```
mlp.fit(X_tr, y_sex_tr)
p_sex    = mlp.predict_proba(X_val)[:,1]
hold_sex = roc_auc_score(y_sex_val, p_sex)

# ADHD Model
cv_adhd   = cross_val_score(mlp, X_tr, y_adhd_tr, cv=5, scoring='roc_auc')
mlp.fit(X_tr, y_adhd_tr)
p_adhd    = mlp.predict_proba(X_val)[:,1]
hold_adhd = roc_auc_score(y_adhd_val, p_adhd)

print(f"PCA MLP Sex  5-fold CV AUC: {cv_sex.mean():.3f}")
print(f"PCA MLP Sex  Hold-out AUC:  {hold_sex:.3f}\n")
print(f"PCA MLP ADHD 5-fold CV AUC: {cv_adhd.mean():.3f}")
print(f"PCA MLP ADHD Hold-out AUC:  {hold_adhd:.3f}")


import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
import numpy as np
# bar chart
metrics = {
    'ADHD': {},
    'Sex':  {}
}

for name, y_true, y_proba in [
    ('ADHD', y_adhd_val, p_adhd_hold),
    ('Sex',  y_sex_val,  p_sex_hold)
]:
    y_pred = (y_proba >= 0.5).astype(int)
    metrics[name]['Accuracy']  = accuracy_score(y_true, y_pred)
    metrics[name]['Precision'] = precision_score(y_true, y_pred, zero_division=0)
    metrics[name]['Recall']    = recall_score(y_true, y_pred, zero_division=0)
    metrics[name]['F1-score']  = f1_score(y_true, y_pred, zero_division=0)

labels = ['Accuracy', 'Precision', 'Recall', 'F1-score']
x = np.arange(len(labels))
width = 0.35

fig, ax = plt.subplots(figsize=(7,4))
adhd_vals = [metrics['ADHD'][lab] for lab in labels]
sex_vals  = [metrics['Sex'][lab]  for lab in labels]

ax.bar(x - width/2, adhd_vals, width, label='ADHD')
ax.bar(x + width/2, sex_vals,  width, label='Sex')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.set_ylabel('Score')
ax.set_title('Threshold-based Metrics (threshold=0.5)')
ax.set_ylim(0,1)
ax.legend()
ax.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# ROC Curve
fpr_ad, tpr_ad, _ = roc_curve(y_adhd_val, p_adhd_hold)
fpr_sx, tpr_sx, _ = roc_curve(y_sex_val,  p_sex_hold)
roc_auc_ad = auc(fpr_ad, tpr_ad)
roc_auc_sx = auc(fpr_sx, tpr_sx)

plt.figure(figsize=(6,6))
plt.plot(fpr_ad, tpr_ad, label=f'ADHD ROC (AUC = {roc_auc_ad:.2f})', linewidth=2)
plt.plot(fpr_sx, tpr_sx, label=f'Sex ROC (AUC = {roc_auc_sx:.2f})', linewidth=2)
plt.plot([0,1], [0,1], linestyle=':', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('MLP PCA Model ROC Curves')
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()


- Encoded+functional in with Feature Engineering

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.feature_selection import SelectKBest, f_classif

# custom Feature Engineering transformer
class FeatureEngineer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        X = X.copy()
        # bin EHQ_EHQ_Total into three quantiles and one-hot encode
        X['EHQ_bin'] = pd.qcut(X['EHQ_EHQ_Total'], q=3, labels=False, duplicates='drop')
        X = pd.get_dummies(X, columns=['EHQ_bin'], prefix='EHQ_bin')
        # Sum_APQ & interaction term
        apq_cols = [c for c in X.columns if c.startswith('APQ_P_APQ_P_')]
        X['Sum_APQ'] = X[apq_cols].sum(axis=1)
        X['CVxAPQ'] = X['ColorVision_CV_Score'] * X['Sum_APQ']

        # polynomial expansion (EHQ, CV, Sum_APQ)
        poly = PolynomialFeatures(degree=2, include_bias=False)
        feats = poly.fit_transform(X[['EHQ_EHQ_Total','ColorVision_CV_Score','Sum_APQ']])
        names = poly.get_feature_names_out()
        df_poly = pd.DataFrame(feats, columns=names, index=X.index)
        X = pd.concat([X, df_poly], axis=1)

        return X.values

# split dataset
df = pd.read_csv('/content/drive/MyDrive/datathon/Datathon/Processed new/encoded/include functional/
X = df.drop(columns=['participant_id','ADHD_Outcome','Sex_F'])
y = df['ADHD_Outcome']

X_tr, X_val, y_tr, y_val = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42)

# Build a Pipeline
pipe = Pipeline([
    ('fe',     FeatureEngineer()),
    ('scaler', StandardScaler()),
    ('sel',    SelectKBest(score_func=f_classif)),
    ('clf',    MLPClassifier(max_iter=1000, random_state=42))
])

param_grid = {
    'sel__k':                 [10, 20, 30],
    'clf__hidden_layer_sizes': [(50,), (100,), (50,50)],
    'clf__alpha':             [1e-4, 1e-3],
    'clf__learning_rate_init':[1e-3, 1e-2],}

grid = GridSearchCV(
    pipe,
    param_grid,
    scoring='roc_auc',
    cv=5,
    n_jobs=-1,
    verbose=1)
```

```
grid.fit(X_tr, y_tr)

print(">>> FE MLP ADHD CV AUC:", grid.best_score_)
print(">>> FE MLP ADHD Params:",  grid.best_params_)

# test ADHD Model
p_val = grid.predict_proba(X_val)[:,1]
print(">>> FE MLP ADHD Hold-out AUC:", roc_auc_score(y_val, p_val))

# same process for sex model
df2 = pd.read_csv('/content/drive/MyDrive/datathon/Datathon/Processed new/encoded/no functional/trai
X2 = df2.drop(columns=['participant_id','ADHD_Outcome','Sex_F'])
y2 = df2['Sex_F']

X2_tr, X2_val, y2_tr, y2_val = train_test_split(
    X2, y2, test_size=0.2, stratify=y2, random_state=42)

grid2 = GridSearchCV(
    pipe,
    param_grid,
    scoring='roc_auc',
    cv=5,
    n_jobs=-1,
    verbose=1)
grid2.fit(X2_tr, y2_tr)

print(">>> FE MLP Sex CV AUC:", grid2.best_score_)
print(">>> FE MLP Sex Params:",  grid2.best_params_)
p2_val = grid2.predict_proba(X2_val)[:,1]
print(">>> FE MLP Sex Hold-out AUC:", roc_auc_score(y2_val, p2_val))


import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# plot bar chart
threshold = 0.5
y_adhd_pred = (p_adhd_hold >= threshold).astype(int)
y_sex_pred  = (p_sex_hold  >= threshold).astype(int)

metrics = {
    'Accuracy':  [accuracy_score(y_adhd_val, y_adhd_pred),
                  accuracy_score(y_sex_val,  y_sex_pred)],
    'Precision': [precision_score(y_adhd_val, y_adhd_pred),
                  precision_score(y_sex_val,  y_sex_pred)],
    'Recall':    [recall_score(y_adhd_val, y_adhd_pred),
                  recall_score(y_sex_val,  y_sex_pred)],
    'F1-score':  [f1_score(y_adhd_val, y_adhd_pred),
                  f1_score(y_sex_val,  y_sex_pred)],
}

tasks = ['ADHD', 'Sex']
x = np.arange(len(tasks))
width = 0.20

fig, ax = plt.subplots()
for i, (name, vals) in enumerate(metrics.items()):
    ax.bar(x + i*width, vals, width, label=name)

ax.set_xticks(x + width*1.5)
ax.set_xticklabels(tasks)
ax.set_ylabel('Score')
ax.set_title('Threshold-based Metrics (threshold=0.5)')
ax.legend()
plt.show()


# plot ROC
from sklearn.metrics import roc_curve, roc_auc_score

fpr_adhd, tpr_adhd, _ = roc_curve(y_adhd_val, p_adhd_hold)
fpr_sex,  tpr_sex,  _ = roc_curve(y_sex_val,  p_sex_hold)

fig, ax = plt.subplots()
ax.plot(fpr_adhd, tpr_adhd,
        label=f'ADHD ROC (AUC = {roc_auc_score(y_adhd_val, p_adhd_hold):.2f})')
ax.plot(fpr_sex, tpr_sex,
        label=f'Sex   ROC (AUC = {roc_auc_score(y_sex_val,  p_sex_hold):.2f})')
ax.plot([0, 1], [0, 1], linestyle='--', label='Chance')
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title('FE ROC Curves')
ax.legend()
plt.show()


## MLP Test Comparison

| Pipeline                      | Sex CV AUC | Sex Hold-out AUC | ADHD CV AUC | ADHD Hold-out AUC |
|-------------------------------|------------|------------------|-------------|-------------------|
| **Baseline MLP**              | 0.579      | 0.540              | 0.730       | 0.793            |
| **PCA + MLP**                 | 0.611      | 0.531              | 0.709       | 0.736            |
| **FE + MLP**                  | 0.548      | 0.452              | 0.784       | 0.810            |


### Summary

- **Baseline MLP**
  - Utilized only original Meta features (excluding Functional). Sex task performed poorly (CV 0.579

- **PCA + MLP**
  - Functional data was reduced via PCA and combined with Meta features, significantly improving Sex

- **Feature Engineering + MLP**
  - Custom feature engineering (binning, interactions, polynomial expansions) followed by SelectKBes



### Next Step
- Construct new candidate feature sets using the Intersection and Union strategies.
- Feed these candidate feature sets into an MLP model for modeling and comparison, further validatin


# Feature Selection Procedure for Sex Binary Taskencoded + function not in
import pandas as pd
from sklearn.feature_selection import SelectKBest, f_classif
df = pd.read_csv('/Users/chenliu/Desktop/CU Files/5205/project/Processed new/encoded/no functional/t
df2 = pd.read_csv('/Users/chenliu/Desktop/CU Files/5205/project/Processed new/encoded/no functional/
X2 = df.drop(columns=['participant_id','ADHD_Outcome','Sex_F'])
y2 = df2['Sex_F']

# split into train / hold-out
from sklearn.model_selection import train_test_split
X2_tr, X2_val, y2_tr, y2_val = train_test_split(
    X2, y2, test_size=0.2, stratify=y2, random_state=42)

# Univariate selection: top 20 by ANOVA F-test
skb = SelectKBest(f_classif, k=20).fit(X2_tr, y2_tr)
top20_skb = pd.Series(skb.scores_, index=X2.columns).nlargest(20)
print("SelectKBest Top 20:\n", top20_skb)

# Tree-based importance: top 20 from Random Forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=200, random_state=42).fit(X2_tr, y2_tr)
imp = pd.Series(rf.feature_importances_, index=X2.columns).nlargest(20)
```

```
print("RandomForest Top 20:\n", imp)


# Intersection Experiment
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score

# Intersection of the two top-20 lists
skb_top = top20_skb.index
rf_top  = imp.index
sex_feats_inter = skb_top.intersection(rf_top).tolist()
print("Intersection features", sex_feats_inter)

# Build a Pipeline
pipe_inter = Pipeline([
    ('scale', StandardScaler()),
    ('sel',   SelectKBest(f_classif)),
    ('clf',   MLPClassifier(max_iter=1000, random_state=42))
])

# Grid-search only that many features, plus a few MLP hyper-parameters
param_grid = {
    'sel__k': [len(sex_feats_inter)],
    'clf__hidden_layer_sizes': [(50,),(100,),(50,50)],
    'clf__alpha': [1e-4, 1e-3, 1e-2],
    'clf__learning_rate_init': [1e-3, 1e-2],
}
grid_inter = GridSearchCV(
    pipe_inter, param_grid, scoring='roc_auc',
    cv=5, n_jobs=-1, verbose=1, error_score='raise')
grid_inter.fit(X2_tr[sex_feats_inter], y2_tr)

# test
print(">>> Intersection Sex CV AUC:", grid_inter.best_score_)
print(">>> Intersection Sex Params:", grid_inter.best_params_)

p_inter = grid_inter.predict_proba(X2_val[sex_feats_inter])[:,1]
print(">>> IntersectionSex Hold-out AUC:", roc_auc_score(y2_val, p_inter))


# Union experiment
# Union of the two top-20 lists
```

```
skb_top = top20_skb.index
rf_top  = imp.index
sex_feats_union = skb_top.union(rf_top).tolist()
print("Union features counts", len(sex_feats_union))

# Build a Pipeline
pipe_union = Pipeline([
    ('scale', StandardScaler()),
    ('sel',   SelectKBest(f_classif)),
    ('clf',   MLPClassifier(max_iter=1000, random_state=42))
])

# Grid-search over different k values
param_grid = {
    'sel__k': [5, 10, 15, 20],
    'clf__hidden_layer_sizes': [(50,),(100,),(50,50)],
    'clf__alpha': [1e-4, 1e-3, 1e-2],
    'clf__learning_rate_init': [1e-3, 1e-2],
}
grid_union = GridSearchCV(
    pipe_union, param_grid, scoring='roc_auc',
    cv=5, n_jobs=-1, verbose=1, error_score='raise')

grid_union.fit(X2_tr[sex_feats_union], y2_tr)

# test
print(">>> Union Sex CV AUC:", grid_union.best_score_)
print(">>> Union Sex Params:", grid_union.best_params_)

p_union = grid_union.predict_proba(X2_val[sex_feats_union])[:,1]
print(">>> Union Sex Hold-out AUC:", roc_auc_score(y2_val, p_union))


## Test Results Comparison

| Pipeline                             | CV Sex AUC | Hold-out Sex AUC |
|--------------------------------------|------------|------------------|
| **Baseline MLP**                     | 0.579      | 0.540            |
| **Intersection FE + MLP**            | 0.599      | 0.554            |
| **Union FE + MLP**                   | 0.571      | 0.467            |


### Summary

- **Intersection FE + MLP**:
  Achieved a slight edge over the baseline (CV 0.599 vs 0.579; Hold-out 0.554 vs 0.540), but the imp
```

- **Union FE + MLP**:
  Underperformed the baseline (CV 0.571 vs 0.579; Hold-out 0.467 vs 0.540), suggesting that combinin

- **High variance**:
  Noticeable drop from CV to Hold-out (e.g. Intersection: 0.599 → 0.554) indicates instability acros

- **Selection pitfalls**:
  Both univariate F-test and tree-based importances struggle to capture complex, higher-order nonlin

- **Signal loss/distortion**:
  Aggressive pruning either filters out or distorts crucial sex-related signals, leading to worse pe

### Next Step

- **Switch to LightGBM** for Sex classification using the **full original feature set**.
- LightGBM naturally models high-order nonlinear interactions across all features, retains all poten

# Random Forest

- not encoded and include functional data
- Baseline Random Forest Model

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
from sklearn.ensemble       import RandomForestClassifier
from sklearn.metrics        import roc_auc_score
from sklearn.model_selection import train_test_split

# Read data
df = pd.read_csv('/content/drive/MyDrive/datathon/Datathon/Processed new/not encoded/include functio

X = df.drop(columns=['participant_id','Sex_F','ADHD_Outcome'])
y_adhd = df['ADHD_Outcome']
y_sex  = df['Sex_F']

# Baseline: RF on ADHD Model
X_tr, X_val, y_tr, y_val = train_test_split(
    X, y_adhd,
    test_size=0.2,
    stratify=y_adhd,
    random_state=42)
```

```
rf = RandomForestClassifier(
    n_estimators=200,
    max_depth=8,
    class_weight='balanced',
    random_state=42)

rf.fit(X_tr, y_tr)
p_val = rf.predict_proba(X_val)[:,1]
print("Baseline RF ADHD AUC:", roc_auc_score(y_val, p_val))

# Repeat for Sex Model
X_tr2, X_val2, y_tr2, y_val2 = train_test_split(
    X, y_sex,
    test_size=0.2,
    stratify=y_sex,
    random_state=42)

rf2 = RandomForestClassifier(
    n_estimators=200,
    max_depth=8,
    class_weight='balanced',
    random_state=42)

rf2.fit(X_tr2, y_tr2)
p_val2 = rf2.predict_proba(X_val2)[:,1]
print("Baseline RF Sex AUC:", roc_auc_score(y_val2, p_val2))


import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
fpr_adhd, tpr_adhd, _ = roc_curve(y_val, p_val)
roc_auc_adhd        = auc(fpr_adhd, tpr_adhd)

fpr_sex, tpr_sex, _ = roc_curve(y_val2, p_val2)
roc_auc_sex         = auc(fpr_sex, tpr_sex)

# plot ROC
plt.figure()
plt.plot(fpr_adhd, tpr_adhd, label=f'ADHD ROC (AUC = {roc_auc_adhd:.2f})')
plt.plot(fpr_sex,  tpr_sex,  label=f'Sex ROC   (AUC = {roc_auc_sex:.2f})')
plt.plot([0,1],   [0,1],    '--',      label='Chance')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Baseline Curves for Random Forest')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()


## Objective
After running the original Baseline RF, we observed:
- The raw Meta+FC features produce very weak signal for predicting **Sex** (RF AUC  0.53).
- Predicting **ADHD** is acceptable (RF AUC  0.82).

To address this, we applied our **feature-engineered version** of the data:
1. **Meta features**: Quantile-based binning on ‘EHQ_EHQ_Total‘ → one-hot encoding
2. **Interaction**: Sum of all ‘APQ_P_APQ_P_‘ scores + interaction term ‘ColorVision_CV_Score * Sum_
3. **Polynomial expansion**: Degree-2 features (including cross-terms) on ‘[EHQ_EHQ_Total, ColorVisi
4. **SelectKBest**: For **each task** (ADHD vs Sex) pick the top 20 features by ‘f_classif‘
5. On **each 20-feature subset**, perform an 80/20 stratified split, train a balanced RandomForest,

---

import pandas as pd
from sklearn.model_selection   import train_test_split
from sklearn.preprocessing    import KBinsDiscretizer, PolynomialFeatures
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.ensemble         import RandomForestClassifier
from sklearn.metrics          import roc_auc_score

X0      = df.drop(columns=['participant_id','Sex_F','ADHD_Outcome'])
y_adhd  = df['ADHD_Outcome']
y_sex   = df['Sex_F']

# Feature Engineering
# EHQ bining → one-hot
kb           = KBinsDiscretizer(n_bins=3, encode='onehot-dense', strategy='quantile')
ehq_bins     = kb.fit_transform(X0[['EHQ_EHQ_Total']])
X0[['EHQ_low','EHQ_mid','EHQ_high']] = ehq_bins

# APQ sum and intersection
apq_cols     = [c for c in X0.columns if c.startswith('APQ_P_APQ_P_')]
X0['Sum_APQ']    = X0[apq_cols].sum(axis=1)
X0['CVxAPQ_sum'] = X0['ColorVision_CV_Score'] * X0['Sum_APQ']

# polynomial expansion
poly         = PolynomialFeatures(degree=2, include_bias=False)
poly_in      = X0[['EHQ_EHQ_Total','ColorVision_CV_Score','Sum_APQ']]
poly_feats   = poly.fit_transform(poly_in)
poly_names   = poly.get_feature_names_out(poly_in.columns)
```

```
X0[poly_names] = poly_feats

# Feature Selection
def fe_rf_evaluate(X_full, y, name):
    # Select Top20
    selector = SelectKBest(f_classif, k=20)
    X_sel    = selector.fit_transform(X_full, y)
    cols     = X_full.columns[selector.get_support()].tolist()

    # Split train and test 80/20
    X_tr, X_val, y_tr, y_val = train_test_split(
        pd.DataFrame(X_sel, columns=cols),
        y,
        test_size=0.2,
        stratify=y,
        random_state=42)

    # Training Model
    rf = RandomForestClassifier(
        n_estimators=200,
        max_depth=8,
        class_weight='balanced',
        random_state=42)

    rf.fit(X_tr, y_tr)
    p_val = rf.predict_proba(X_val)[:,1]
    auc   = roc_auc_score(y_val, p_val)

    print(f"{name} (FE) selected 20 features:", cols)
    print(f"{name} (FE) RF AUC: {auc:.3f}\n")

fe_rf_evaluate(X0, y_adhd, "ADHD")
fe_rf_evaluate(X0, y_sex,  "Sex")


import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
fpr_adhd, tpr_adhd, _ = roc_curve(y_val, p_val)
roc_auc_adhd        = auc(fpr_adhd, tpr_adhd)

fpr_sex, tpr_sex, _ = roc_curve(y_val2, p_val2)
roc_auc_sex         = auc(fpr_sex, tpr_sex)

# plot ROC
plt.figure()
plt.plot(fpr_adhd, tpr_adhd, label=f'ADHD ROC (AUC = {roc_auc_adhd:.2f})')
plt.plot(fpr_sex,  tpr_sex,  label=f'Sex ROC   (AUC = {roc_auc_sex:.2f})')
plt.plot([0,1],   [0,1],    '--',      label='Chance')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC FE Curves for Random Forest Baseline')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()


## Random Forest Test Comparison

| Pipeline                  | Sex AUC | ADHD AUC |
|---------------------------|--------:|---------:|
| Baseline RF (all Meta+FC) |   0.592 |    0.803 |
| Feature-Engineered + RF   |   0.752 |    0.850 |




- **ADHD**: Feature engineering and targeted feature selection improved AUC from ~0.803 → **0.850**.
- **Sex**: Major boost from AUC ~0.592 → **0.752**, indicating that our engineered Meta-only feature

**Next Steps:**
- Consider cross-validation and hyperparameter tuning on the 20-feature subsets.


import pandas as pd
from sklearn.model_selection   import StratifiedKFold, GridSearchCV
from sklearn.pipeline          import Pipeline
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.ensemble         import RandomForestClassifier
from sklearn.metrics          import make_scorer, roc_auc_score

X0     = df.drop(columns=['participant_id','Sex_F','ADHD_Outcome'])
y_adhd = df['ADHD_Outcome']
y_sex  = df['Sex_F']

def build_search(y, name):
    pipe = Pipeline([
        # Select Top20
        ('select', SelectKBest(f_classif, k=20)),
        # Random Forest Model
        ('rf',     RandomForestClassifier(class_weight='balanced', random_state=42))])
    param_grid = {
        'rf__n_estimators': [100, 200, 500],
        'rf__max_depth':    [5, 8, 12, None],
        'rf__min_samples_leaf': [1, 3, 5]}

    # try cross- validation
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    search = GridSearchCV(
        pipe, param_grid,
        scoring=make_scorer(roc_auc_score, needs_proba=True),
        cv=cv, verbose=1, n_jobs=-1)

    print(f"\n>>> Starting hyperparameter search for {name} ...")
    search.fit(X0, y)
    print(f"{name} best AUC: {search.best_score_:.3f}")
    print(f"{name} best params: {search.best_params_}\n")
    return search

search_adhd = build_search(y_adhd, 'ADHD')
search_sex  = build_search(y_sex,  'Sex')


import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import KBinsDiscretizer, PolynomialFeatures
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score

def full_fe_cv_mean_std(df: pd.DataFrame,
                        target_col: str,
                        k_features: int = 20,
                        n_splits: int = 5,
                        random_state: int = 42):

    rf_params = {
        'n_estimators': 200,
        'max_depth': 8,
        'class_weight': 'balanced',
        'random_state': random_state}


    X0 = df.drop(columns=['participant_id', 'Sex_F', 'ADHD_Outcome'])
    y  = df[target_col]
```

```
skf = StratifiedKFold(n_splits=n_splits,
                      shuffle=True,
                      random_state=random_state)
aucs = []

for tr_idx, te_idx in skf.split(X0, y):
    # split train / test
    X_tr, X_te = X0.iloc[tr_idx].copy(), X0.iloc[te_idx].copy()
    y_tr, y_te = y.iloc[tr_idx], y.iloc[te_idx]

    # EHQ bining
    kb = KBinsDiscretizer(n_bins=3, encode='onehot-dense', strategy='quantile')
    ehq_tr = kb.fit_transform(X_tr[['EHQ_EHQ_Total']])
    ehq_te = kb.transform(X_te[['EHQ_EHQ_Total']])
    X_tr[['EHQ_low','EHQ_mid','EHQ_high']] = ehq_tr
    X_te[['EHQ_low','EHQ_mid','EHQ_high']] = ehq_te

    # APQ sum & intersection
    apq_cols = [c for c in X_tr.columns if c.startswith('APQ_P_APQ_P_')]
    X_tr['Sum_APQ']    = X_tr[apq_cols].sum(axis=1)
    X_te['Sum_APQ']    = X_te[apq_cols].sum(axis=1)
    X_tr['CVxAPQ_sum'] = X_tr['ColorVision_CV_Score'] * X_tr['Sum_APQ']
    X_te['CVxAPQ_sum'] = X_te['ColorVision_CV_Score'] * X_te['Sum_APQ']

    # Polynomial Expansion
    poly = PolynomialFeatures(degree=2, include_bias=False)
    base = ['EHQ_EHQ_Total','ColorVision_CV_Score','Sum_APQ']
    pf_tr = poly.fit_transform(X_tr[base])
    pf_te = poly.transform(X_te[base])
    names = poly.get_feature_names_out(base)
    X_tr[names] = pf_tr
    X_te[names] = pf_te

    # Feature Selection
    sel = SelectKBest(f_classif, k=k_features)
    sel.fit(X_tr, y_tr)
    X_tr_sel = sel.transform(X_tr)
    X_te_sel = sel.transform(X_te)

    # Train Model
    rf = RandomForestClassifier(**rf_params)
    rf.fit(X_tr_sel, y_tr)
    probs = rf.predict_proba(X_te_sel)[:, 1]
    aucs.append(roc_auc_score(y_te, probs))

mean_auc = np.mean(aucs)
```

```
    std_auc  = np.std(aucs)
    print(f"{target_col}  {n_splits}-fold CV AUC: {mean_auc:.3f} ± {std_auc:.3f}")
    return mean_auc, std_auc

# For ADHD
full_fe_cv_mean_std(df, 'ADHD_Outcome', k_features=20, n_splits=5)

# For Sex_F
full_fe_cv_mean_std(df, 'Sex_F',        k_features=20, n_splits=5)

## Random Forest FE Model Results Comparison

| Task   | Hold-out AUC | 5-Fold CV AUC | Change      |
|--------|--------------|---------------|-------------|
| **ADHD** | 0.850        | 0.793         | Slightly      |
| **Sex**  | 0.752        | 0.589         | Significant drop |



## Conclusion

- **ADHD prediction** remains consistent between a single hold-out split and 5-fold cross-validatior
- **Sex prediction** suffers a notable performance decline under cross-validation (from  0.752 down



## Next Step

-  **Explore additional model families**
   - Try XGBoost, LightGBM, or a small neural network; these can sometimes detect subtle patterns th
```