LEARN MATE

"Empowering Autism, Unlocking Potential"

Team

Tharaki D.H.D
IT21254970

Ruhunuge R.S.D.P
IT21227486

Rupasinghe Y.S
IT21160820

Pehesarani W.K.A
IT21259470

Mr. Samadhi Chathuranga
Supervisor

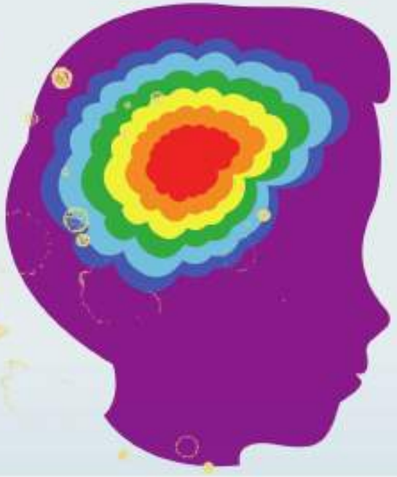
Ms. Thisara Shyamalee
Co-Supervisor

Prof. Hemamali Perera
External Supervisor

Background

ASD (Autism Spectrum Disorder) - A neuro-developmental condition of variable severity with lifelong effects that can be recognized from early childhood,

Affects communication, learning, and behavior of a person



Introduction

- Enhance the learning skills of autistic children using machine learning techniques.
- Provide customized recommendation plans on a weekly basis.
- Capture detailed progress of each child.
- Visualize the child's progress through intuitive graphs and charts using color theory.
- Enable parents and teachers to monitor and review the child's development.

Research Problems

How to enhance autistic children's abilities based on domains of learning with the use of machine learning?

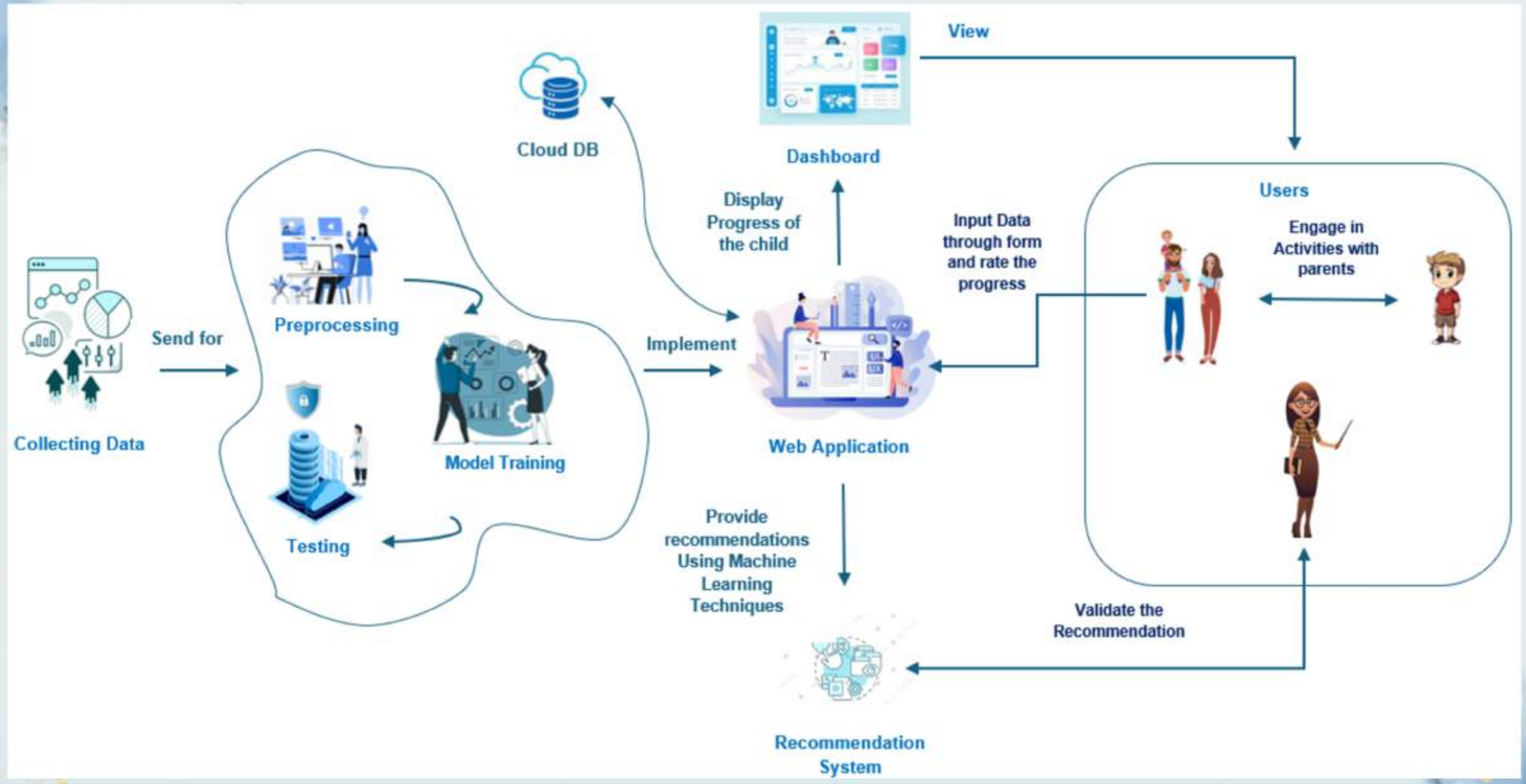
How to create peer-support strategies that foster social interactions and understanding?

Proposed Solutions

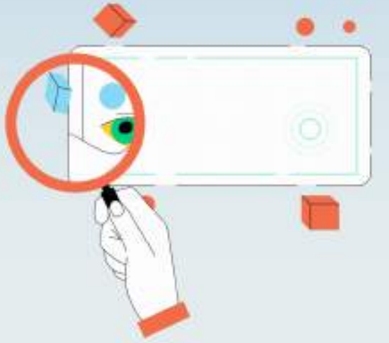
- Integrating Learning Recommendations with Parent-Friendly Dashboards

- Peer Support and Awareness Modules

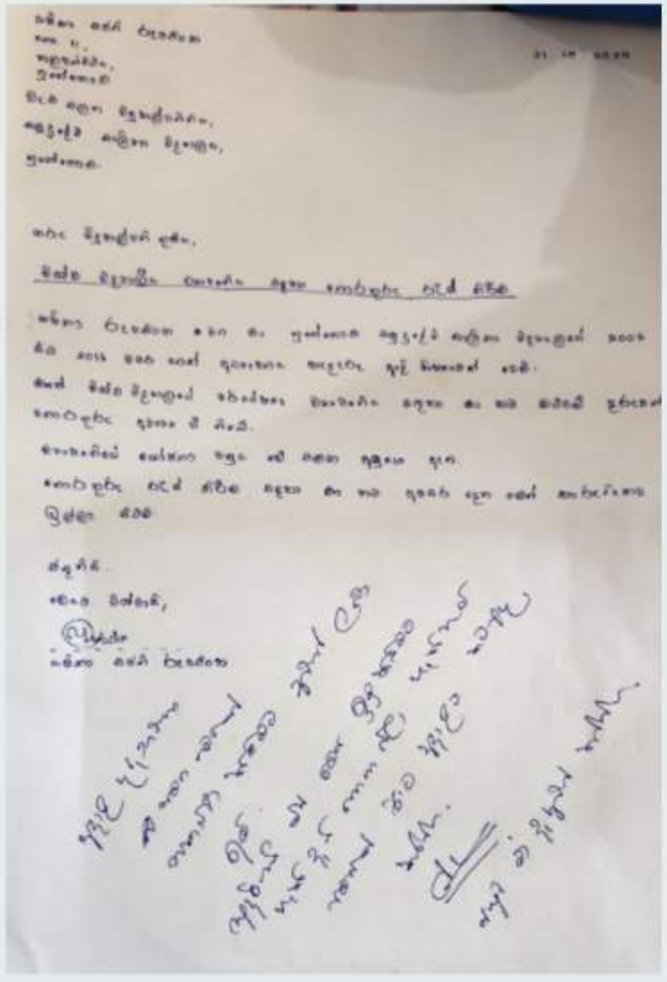
Solution Architecture



Data Collection



- **Google Form Development**
  - Created a Google Form to collect data about autistic children. Link
  - Questions focused on learning domains
- **Expert Guidance**
  - Form structure and questions were designed with incorporated knowledge and reference documents provided by Mr. Lakmal Ponnamperuma
- **Distribution**
  - Distributed the Google Form through schools and learning centers.
    - Ayati Centre - Ragama
    - Chithralane Centre - Narahenpita
    - Ash Alifaa Centre - Kotikawatta
    - MJF Centre - Moratuwa
    - Sunera Padanma- Homagama
    - Oruwala Central School - Oruwala
    - Samudradevi School - Nugegoda
    - Kottawa Darmapala Primary School - Digana



intern.chitralane <intern.chitralane@gmail.com> Wed, Sep 18, 9:41AM

to me

Dear Dilmi,

Your group may visit our institute on either of the following days,
25th September
26th September
27th September

If a later day is required we can consider a possible date next month. I can arrange resource persons and observation and other requirements for you and get appointments with necessary parties once you confirm that you're attending by replying to this email with the most preferred date specified.

Please note that there is a payment to be made which is 5000/= per person for a project which will be charged from you upon arrival.

further details and instructions could be shared upon confirmation from your end.

Best Regards,

Inuri Wijesuriya
Project Officer
Chitralane Welfare Society

Ayati Research <ayatiresearch@gmail.com> Thu, Oct 10, 3:14 PM

to me

Dear Dilmi,

Please note that you need approval from the ERC Kelaniya (attached to the Medical Faculty) to continue your research at Ayati.
https://medicine.kln.ac.lk/prog/erc/

Thank you
Regards
Research Committee

Meta-Cognitive Domain

Tharaki D.H.D

Introduction

- Awareness and comprehension of one's own thought and learning processes.
- Critical for problem-solving and academic success.
- Increased Independence and Self-assurance and enhances daily activities.
- Supports social and personal development, aiding in overcoming obstacles.

Knowledge

Regulations

Background

- Target Problem: Insufficient awareness among peers and parents regarding effective strategies to impart knowledge and establish appropriate regulations for supporting the development of autistic children

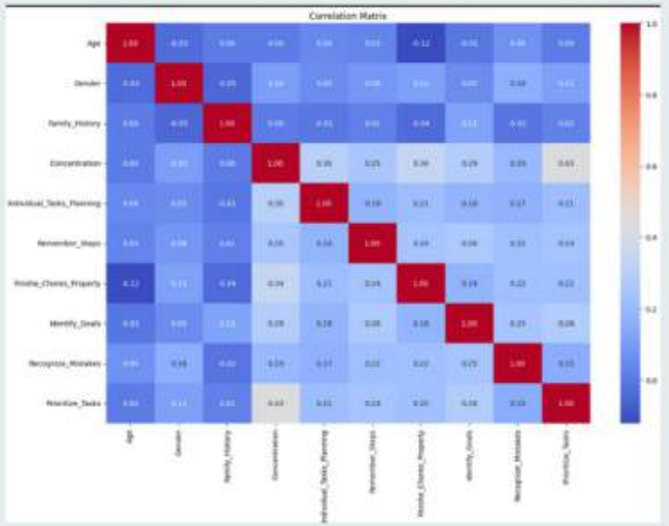
- Proposed Solution: Use clustering methods to categorize autistic children and provide customized activity lists, to be done with the help of parents and caregivers.

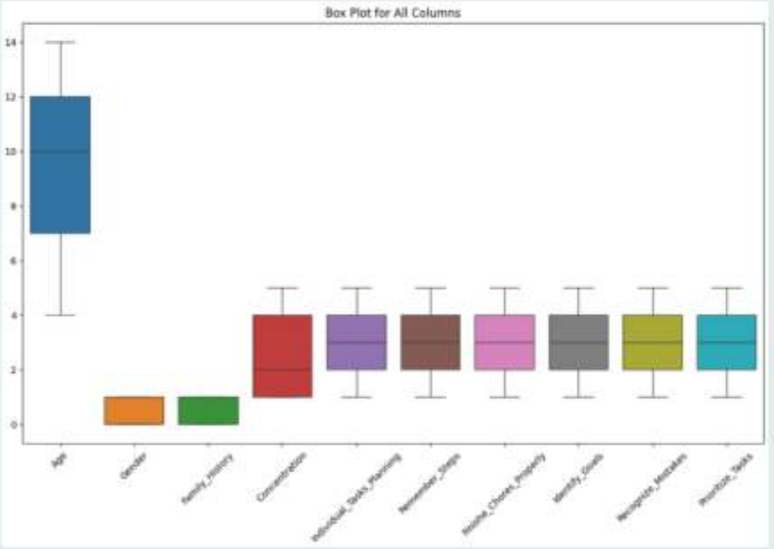
Dataset

- Demographic data
  - Age, Gender, Family_History
- Concentration
  - Ability to maintain focus on a task or activity.
- Individual_Tasks_Planning
  - Capacity to independently organize and plan tasks.
- Remember_Steps
  - Ability to recall and follow a sequence of steps for tasks.
- Finish_Chores_Properly
  - Effectiveness in completing chores or assigned tasks accurately.
- Identify_Goals
  - Capability to set and understand personal or task-specific objectives.
- Recognize_Mistakes
  - Awareness and acknowledgment of errors during task execution.
- Prioritize_Tasks
  - Skill in organizing tasks by importance or urgency.

Data Analysis

```python
# Define mappings for encoding
mappings = {
    'Gender': {'Male': 0, 'Female': 1},
    'Family_History': {'Yes': 1, 'No': 0},
}

def apply_mapping_encoding(df, mappings):
    for column, mapping in mappings.items():
        if column in df.columns:
            df[column] = df[column].map(mapping)
    return df

df1 = apply_mapping_encoding(df, mappings)
```



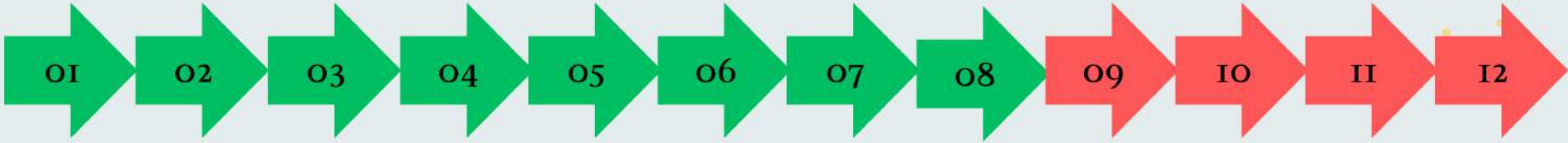



```python
# Function to handle missing values
def handle_missing_values(df):
    for column in df:
        if df[column].dtype == 'object':
            df[column].fillna(df[column].mode()[0], inplace=True)
        else:
            df[column].fillna(df[column].median(), inplace=True)
    return df

df = handle_missing_values(df)
```

Model Building

- Clustering models developed:
  - K-Means
  - DBSCAN
  - Agglomerative Clustering
  - GaussianMixture
- Compare the accuracy and performances of each method and select the best model for the dataset. (using Silhouette Rank, Davies-Bouldin Rank)
- Creating a function to analyze the cluster profiles, calculates severity scores for each cluster, and returns a mapping from cluster index to severity level.

Project Completion



01. Background Search
02. Data Gathering
03. Data Analysis & Pre-processing
04. Model Building
5. Best Model Selection
06.Map the severity levels with clusters
07. Metacognitive Domain Cluster Prediction
08. Customized Recommendations
09. Progress Tracking
10. Building a web app & integrating the models
11. Testing
12. Releasing to the required institutes

Completed 60%

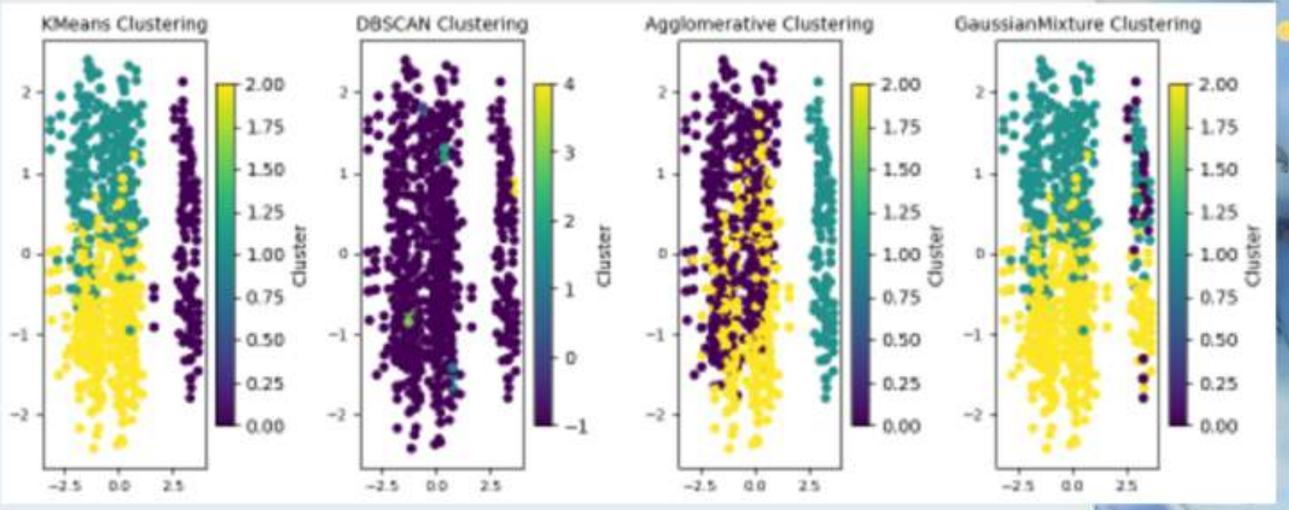
To do 40%

Proof of Work

```python
# Clustering evaluation function
def evaluate_clustering(X_scaled, n_clusters=3):
    results = []
    all_clusters = {}  # Dictionary to store cluster assignments for each method

    # K-Means
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans_clusters = kmeans.fit_predict(X_scaled)
    all_clusters['KMeans'] = kmeans_clusters
    results.append({
        'Method': 'KMeans',
        'Silhouette Score': silhouette_score(X_scaled, kmeans_clusters),
        'Davies-Bouldin Index': davies_bouldin_score(X_scaled, kmeans_clusters)
    })

    # DBSCAN
    dbscan = DBSCAN(eps=0.5, min_samples=5)
    dbscan_clusters = dbscan.fit_predict(X_scaled)
    all_clusters['DBSCAN'] = dbscan_clusters
    results.append({
        'Method': 'DBSCAN',
        'Silhouette Score': silhouette_score(X_scaled, dbscan_clusters) if len(set(dbscan_clusters)) > 1 else -1,
        'Davies-Bouldin Index': davies_bouldin_score(X_scaled, dbscan_clusters) if len(set(dbscan_clusters)) > 1 else -1
    })

    # Agglomerative Clustering
    agglo = AgglomerativeClustering(n_clusters=n_clusters)
    agglo_clusters = agglo.fit_predict(X_scaled)
    all_clusters['Agglomerative'] = agglo_clusters
    results.append({
        'Method': 'Agglomerative',
        'Silhouette Score': silhouette_score(X_scaled, agglo_clusters),
        'Davies-Bouldin Index': davies_bouldin_score(X_scaled, agglo_clusters)
    })

    # Gaussian Mixture
    gm = GaussianMixture(n_components=n_clusters, random_state=42)
    gm_clusters = gm.fit_predict(X_scaled)
    all_clusters['GaussianMixture'] = gm_clusters
    results.append({
        'Method': 'GaussianMixture',
        'Silhouette Score': silhouette_score(X_scaled, gm_clusters),
        'Davies-Bouldin Index': davies_bouldin_score(X_scaled, gm_clusters)
    })

    # Combine cluster assignments into a single array
    combined_clusters = np.column_stack(list(all_clusters.values()))

    return pd.DataFrame(results), combined_clusters
```

Categorize the dataset

```python
[ ] def map_clusters_to_severity(dff, cluster_labels):
        # Add cluster labels to the DataFrame
        dff['Cluster'] = cluster_labels

        # Select only numeric columns (exclude non-numeric like 'Cluster' or categorical data)
        numeric_columns = dff.select_dtypes(include=['number']).columns.tolist()

        # Handle cases where no numeric columns exist
        if not numeric_columns:
            raise ValueError("No numeric columns found in the DataFrame to profile clusters.")

        # Calculate the mean of numeric columns for each cluster
        cluster_profiles = dff.groupby('Cluster')[numeric_columns].mean()

        # Handle missing or NaN values by filling with zeros (optional, based on context)
        cluster_profiles = cluster_profiles.fillna(0)

        # Map clusters to severity levels based on the average value of numeric columns
        # Sort clusters by the overall mean value in descending order
        severity_mapping = cluster_profiles.mean(axis=1).sort_values(ascending=False).index.tolist()

        return severity_mapping
```



Defining user inputs

```
[ ]  def collect_user_input():
        def get_valid_age():
            while True:
                try:
                    age = int(input("Enter Age (4 or older): "))
                    if age >= 4:
                        return age
                    else:
                        print("Age must be 4 or older. Please try again.")
                except ValueError:
                    print("Invalid input. Please enter a valid number for age.")

        def get_valid_rating(prompt):
            while True:
                try:
                    rating = int(input(f"{prompt} (1-5): "))
                    if 1 <= rating <= 5:
                        return rating
                    else:
                        print("Rating must be between 1 and 5. Please try again.")
                except ValueError:
                    print("Invalid input. Please enter a valid number between 1 and 5.")

        # Collect user input
        return {
            'Age': get_valid_age(),
            'Gender': input("Enter Gender (M/F): ").strip().upper(),
            'Family_History': input("Any family member with autism? (yes/no): ").strip().lower(),
            'Concentration': get_valid_rating("Rate Concentration"),
            'Individual_Tasks_Planning': get_valid_rating("Rate Individual Tasks Planning"),
            'Remember_Steps': get_valid_rating("Rate Remembering Steps"),
            'Finishe_Chores_Properly': get_valid_rating("Rate Finishing Chores Properly"),
            'Identify_Goals': get_valid_rating("Rate Can Identifying Goals"),
            'Recognize_Mistakes': get_valid_rating("Rate Can Recognizing Mistakes"),
            'Prioritize_Tasks': get_valid_rating("Rate Prioritizing Tasks")
        }
```

Preprocess Input

```
[ ]  # Function to preprocess the input
     def preprocess_input(user_input):
         # Transform the user input into a suitable format for prediction
         user_input['Gender'] = 1 if user_input['Gender'].lower() == 'M' else 0
         user_input['Family_History'] = 1 if user_input['Family_History'].lower() == 'yes' else 0

         # Example preprocessing: convert to DataFrame (if using pandas)
         return pd.DataFrame([user_input])
```

Provide Recommendations

```
[ ]  # Function to assign user to a cluster and provide recommendations
     def assign_cluster_and_recommend(scaler, kmeans, severity_mapping, user_input):
         # Preprocess the user input
         new_user_data = preprocess_input(user_input)
         new_user_data_scaled = scaler.transform(new_user_data)

         # Predict the cluster for the new user
         assigned_cluster = kmeans.predict(new_user_data_scaled)[0]

         # Map the cluster to severity
         severity = severity_mapping[assigned_cluster]
         print(f"\nThe child is assigned to cluster: {assigned_cluster}")

         # Provide personalized recommendations based on the cluster and user input
         recommended_activities = recommend_activities(assigned_cluster, user_input)

         print("\n\nRecommended activities for the new user:")
         for activity in recommended_activities:
             print(f"- {activity}")
```

```
[ ]  # Dynamically map clusters to severity levels
     severity_mapping = map_clusters_to_severity(dff, best_clusters)

     # Collect user input
     user_input = collect_user_input()

     # Assign cluster and recommend activities
     assign_cluster_and_recommend(scaler, kmeans, severity_mapping, user_input)
```

Cognitive Domain

Rupasinghe Y. S.

Introduction

- Development of intellectual skills such as critical thinking, problem solving and creating a knowledge base.

- 6 levels in cognitive domain:
  - Knowledge
  - Comprehension
  - Application
  - Analysis
  - Synthesis
  - Evaluation

Cognitive Skills

Problem-solving Skills

Knowledge Base

Error Monitoring Skills

Background

- Target Problem: Peers and parents face difficulties when effectively interacting with children with ASD due to insufficient knowledge regarding effective strategies and establish appropriate regulations for supporting the development of autistic children

- Proposed Solution: Use classification models to categorize autistic children and provide customized activity lists, to be done with the help of parents and caregivers.

Dataset

- Features:
  - Demographic : Gender, Age, Family history (If a family member has ASD)
  - Other features:
    1. Problem Solving
    2. Visual Learning Preference
    3. Response to Guidance
    4. Task Independence
    5. Object Identification
    6. Error Correction
- Target variable: Cognitive Level; mild, moderate & severe
- Data volume:
  - Total records: 347
  - Number of features/columns: 08 features

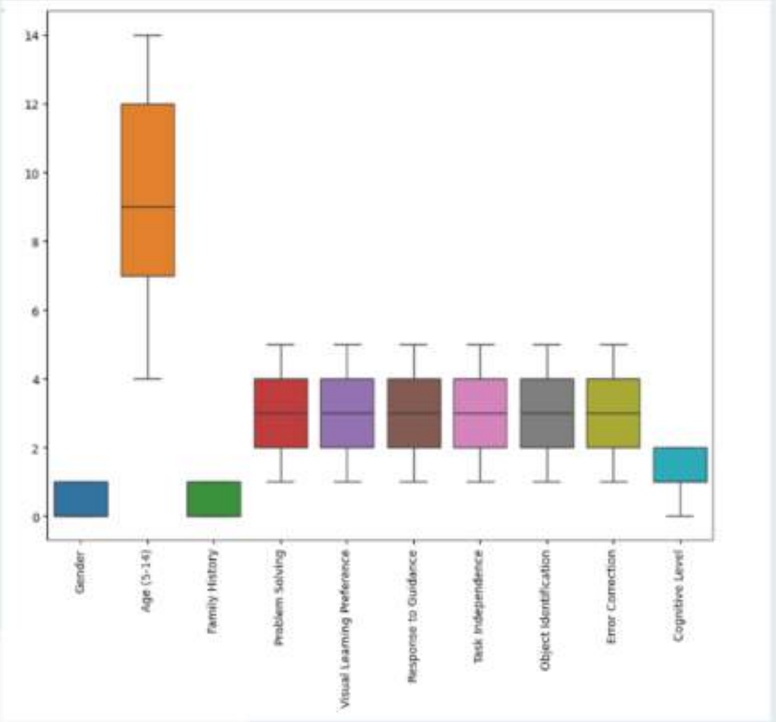
	A	B	C	D	E	F	G	H	I	J
1	Gender	Age (5-14)	Family History	Problem Solving	Visual Learning Preference	Response to Guidance	Task Independence	Object Identification	Error Correction	Cognitive Leve
2	Female	8	No	4	4	3	3	2	1	Moderate
3	Female	14	Yes	2	3	1	3	2	2	Severe
4	Female	10	No	1	1	1	3	2	1	Severe
5	Male	12	No	1	4	2	3	1	1	Severe
6	Male	13	No	4	4	3	5	5	3	Mild
7	Male	12	Yes	2	3	3	3	3	3	Moderate
8	Male	7	Yes	1	3	1	2	2	4	Severe
9	Female	12	No	2	4	5	3	5	5	Mild
10	Male	6	Yes	4	2	5	4	4	4	Mild
11	Male	8	Yes	4	4	2	4	5	4	Mild
12	Female	13	Yes	5	2	1	3	2	2	Severe
13	Male	12	No	3	2	1	3	2	2	Severe
14	Male	13	Yes	1	1	3	1	4	3	Severe
15	Male	8	No	3	3	4	2	2	3	Moderate

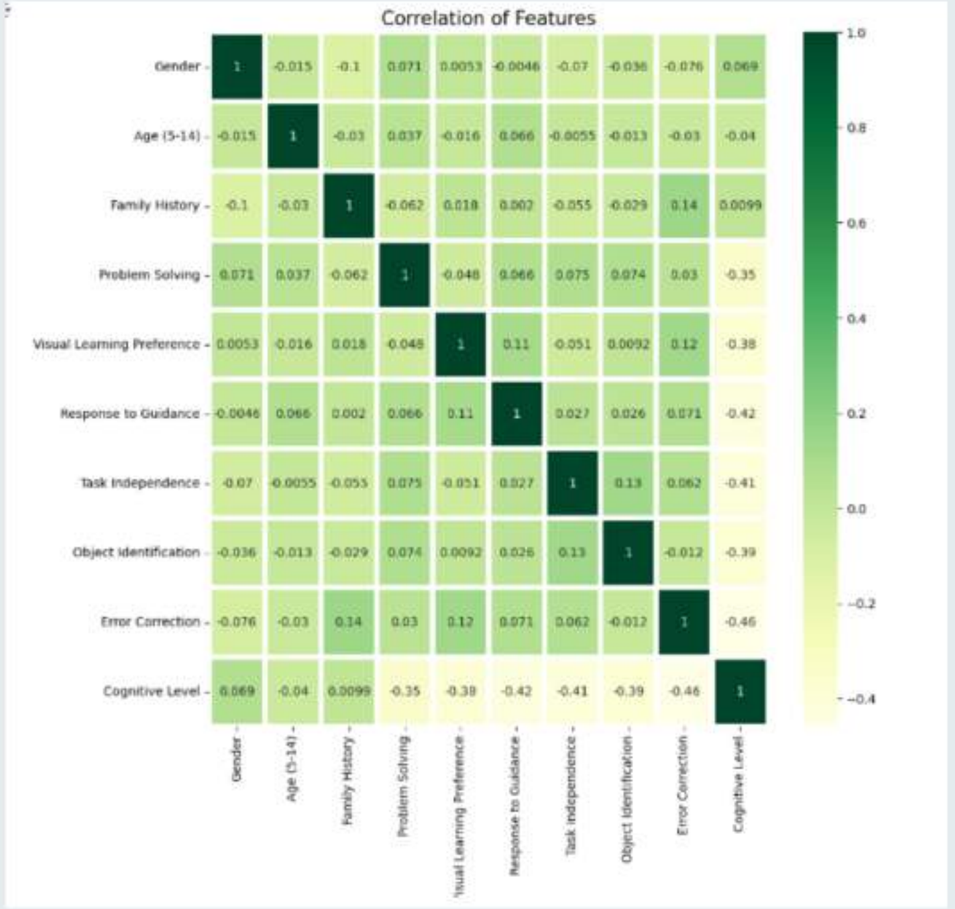
Data Analysis

```
#check for null valuess
df.isnull().sum().sort_values(ascending=False)
```

	0
Gender	0
Age (5-14)	0
Family History	0
Problem Solving	0
Visual Learning Preference	0
Response to Guidance	0
Task Independence	0
Object Identification	0
Error Correction	0
Cognitive Level	0

dtype: int64






```
#encode categorical variables

mappings = {
    "Gender": {"Male": 0, "Female": 1},
    "Family History": {"Yes": 1, "No": 0},
    "Cognitive Level": {"Mild": 0, "Moderate": 1, "Severe": 2}
}

# Apply mappings to the DataFrame
for column, mapping in mappings.items():
    if column in df.columns:  # Check if the column exists in your DataFrame
        df[column] = df[column].map(mapping)
```

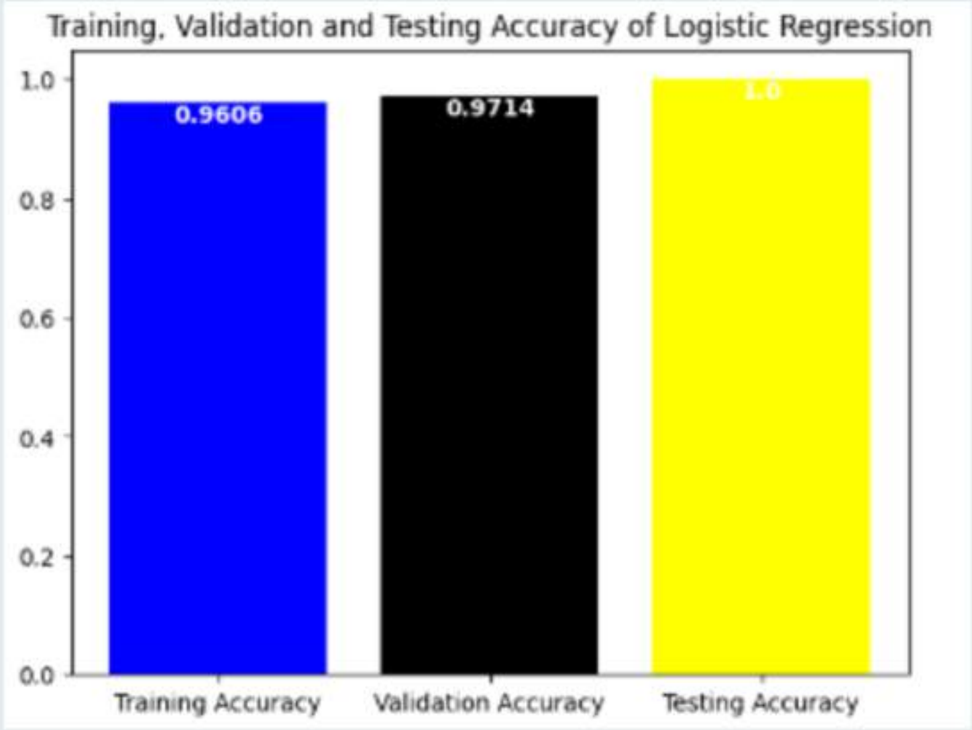
Model Building

- Classification models used:
  - Random Forest Classifier
  - Logistic Regression Model
  - Support Vector Machine
  - XGBoost Model
- Split the dataset in to train, validation and test sets
- Compare the accuracy, recall, precision and F1-score of each method and select the best model
- Creating a function to analyze to predict the “Cognitive Level” of the child with ASD and provide recommendations according to each class.
  - Class 0 -› Mild
  - Class 1 -› Moderate
  - Class 2 -› Severe

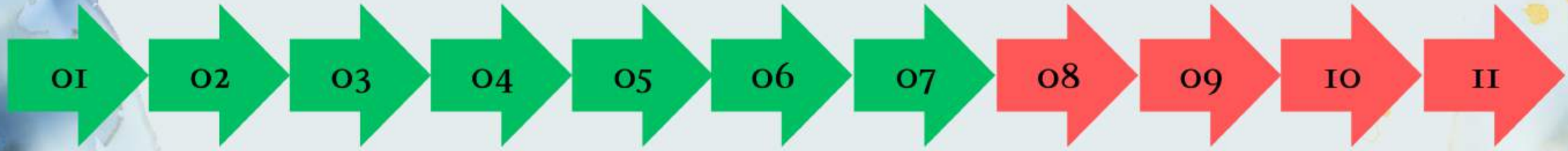
Model Building

- Chose “LOGISTIC REGRESSION MODEL” as the best model

```
Validation Set Performance:
Accuracy: 0.9714285714285714
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       0.93      1.00      0.96        13
           2       1.00      0.94      0.97        16

    accuracy                           0.97        35
   macro avg       0.98      0.98      0.98        35
weighted avg       0.97      0.97      0.97        35


Test Set Performance:
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00         8

    accuracy                           1.00        35
   macro avg       1.00      1.00      1.00        35
weighted avg       1.00      1.00      1.00        35
```



Project completion plan



01. Background Search
02. Data Gathering
03. Data Analysis & Pre-processing
04. Model Building and Validation
05. Best Model Selection
06. Cognitive Level Prediction
07. Customized Recommendations
08. Progress Tracking
09. Building a web app & integrating the models
10. Testing
11. Releasing to the required institutes

Completed (60%)

To do (40%)

Proof Of Work

Predicting the Cognitive Level

```python
def get_valid_CDinput(CDprompt):
    while True:
        user_CDinput = input(CDprompt).strip()
        if not user_CDinput:
            print("Please Enter a Score.")
        elif user_CDinput.isdigit() and 1 <= int(user_CDinput) <= 5:
            return int(user_CDinput)
        else:
            print("Invalid Score. Please Enter a Score between 1 and 5.")

def suggest_CDactivities(CD_level):
    CDactivities = {
        "Mild": [
            "Role-playing Social Scenarios",
            "Routine Building",
            "Social Stories",
            "Group Activities"
        ],
        "Moderate": [
            "Asking questions about the day",
            "Puzzle Solving",
            "Colouring Activities",
            "Collage Activities"
        ],
        "Severe": [
            "Practise Counting",
            "Building Blocks",
            "Matching Activities",
            "Teaching Behaviours",
        ]
    }

    print(f"\nRelevant Activities for {CD_level} Level:")
    activities = CDactivities[CD_level]
    for CDactivity in activities:
        print(f"- {CDactivity}")
    print()
    print()

    return activities
```

```python
def predictUserInput_cognitive_domain_level():
    print("Please enter the following details:")

    gender = input("Gender (Male=1, Female=0): ").strip()
    while gender not in ["0", "1"]:
        print("Invalid Gender. Please enter 1 for Male or 0 for Female.")
        gender = input("Gender (Male=1, Female=0): ").strip()

    age = input("Age: ").strip()
    while not age.isdigit() or int(age) <= 0:
        print("Invalid Age. Please Enter a valid age.")
        age = input("Age: ").strip()

    family_history = input("Family Member with Autism? (Yes=1, No=0): ").strip()
    while family_history not in ["0", "1"]:
        print("Invalid Status. Please enter 1 for Yes or 0 for No.")
        family_history = input("Family Member with Autism? (Yes=1, No=0): ").strip()

    print("\n\nPlease enter the marks for the following questions (1 to 5):")
    problem_solving = get_valid_CDinput("Responding to Problems: ")
    visual_learning_pref = get_valid_CDinput("Visual Learning Preference: ")
    response_to_guidance = get_valid_CDinput("Response to Guidance: ")
    task_independece = get_valid_CDinput("Complete a task independently: ")
    object_identification = get_valid_CDinput("Identifying Objects: ")
    error_correction = get_valid_CDinput("Error Correction Abilities: ")

    # Convert inputs to a NumPy array
    user_data_UserInput = np.array([[int(gender),
                                     int(age),
                                     int(family_history),
                                     problem_solving,
                                     visual_learning_pref,
                                     response_to_guidance,
                                     task_independece,
                                     object_identification,
                                     error_correction]])

    # Predict the level using the logistic regression model
    predicted_CDlevel_UserInput = best_lr_model.predict(user_data_UserInput)

    # Map prediction to a readable level
    engagement_mapping = {2: "Severe", 1: "Moderate", 0: "Mild"}
    predicted_CDlevel = engagement_mapping[predicted_CDlevel_UserInput[0]]
    print("\nPredicted Cognitive Domain Level:", predicted_CDlevel)

    # # Suggest activities
    # activities_for_CDlevel = suggest_CDactivities(predicted_CDlevel)

    return suggest_CDactivities(predicted_CDlevel), predicted_CDlevel
```

```
predictUserInput_cognitive_domain_level()

Please enter the following details:
Gender (Male=1, Female=0): male
Invalid Gender. Please enter 1 for Male or 0 for Female.
Gender (Male=1, Female=0): 1
Age: 6
Family Member with Autism? (Yes=1, No=0): 0


Please enter the marks for the following questions (1 to 5):
Responding to Problems: 1
Make eye contact: 2
Response to Guidance: 2
Caomplete a task independently: 2
Identifying Objects: 2
Error Correction Abilities: 1

Predicted Cognitive Domain Level: Severe

Relevant Activities for Severe Level:
- Practise Counting
- Building Blocks
- Matching Activities
- Teaching Behaviours
```

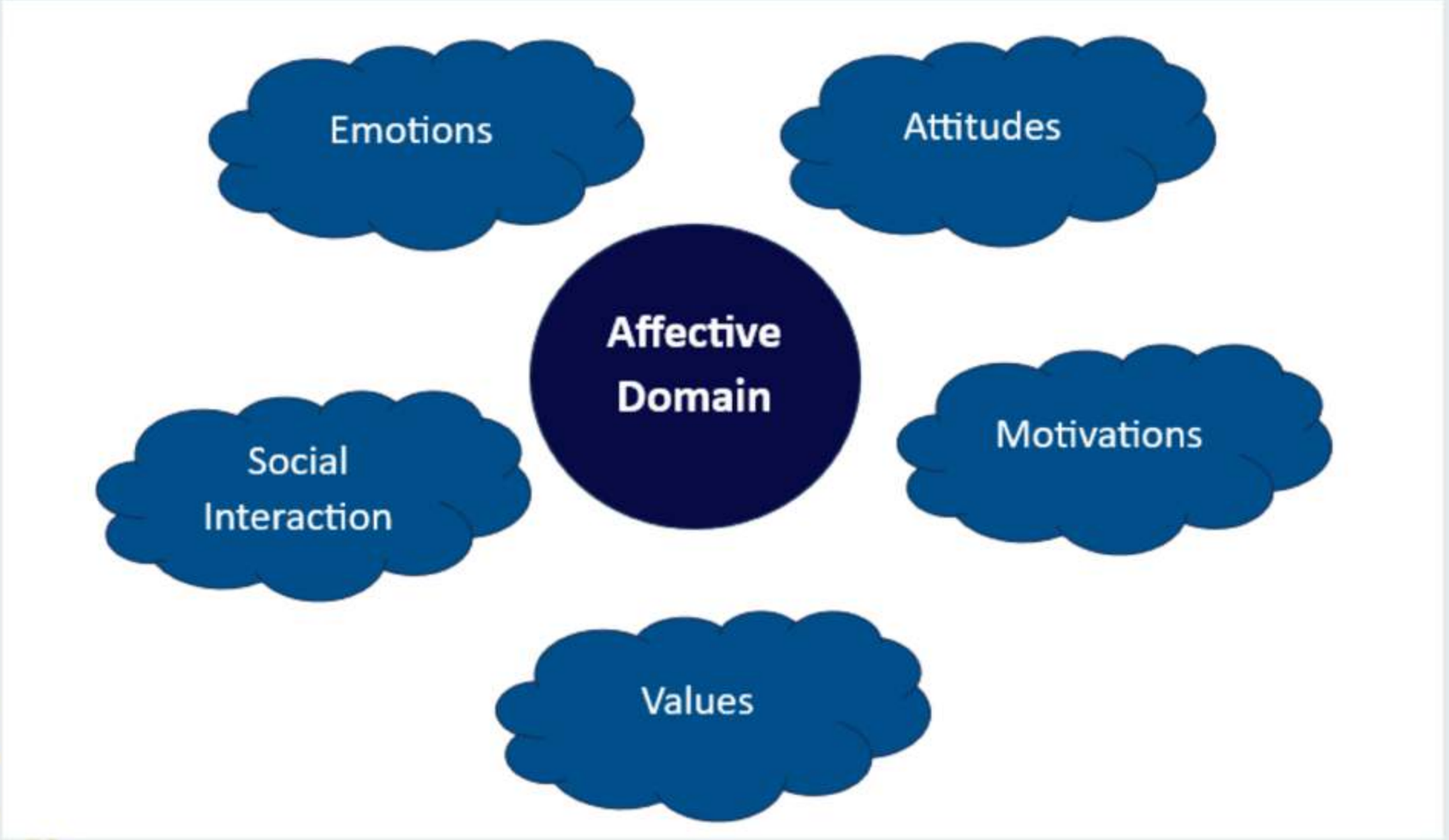
Affective Domain

Ruhunage R.S.D.P

Introduction

- Understanding and managing emotions helps in the construction and maintenance of engaging relations.
- Important for emotional well-being, social life, and adaptive behavior in everyday situations.
- Encourages empathy, regulation of feelings, and belonging to a group, boosting self-confidence.
- Supports the development of positive attitudes and emotional resilience to surmount social challenges.

Affective Domain Skills



Background

- Target Problem: Peers and parents have very limited information concerning the emotional needs and social difficulties of children with autism. This leads to insufficient support in providing them with emotional and social skills.

- Proposed Solution: Employ classification to divided autistic children based on their patterns of emotional and social behavior, and provide customized and emotion-based and social-based activities developed in collaboration with parents, teachers, peers and caregivers, to enhance their emotional and social skills.

Dataset

- Features:
  - Demographic: Gender, Age, Family history (If a family member has ASD)
  - Other features:
    1. Gets along with other children
    2. Make eye contact
    3. Express Feelings appropriately
    4. Show Empathy
    5. Feels happy/angry with reason
    6. Stays Calm
    7. Smiles not less sociably
- Target variable: Affective Domain Level-› mild, moderate & severe
- Data volume:
  - Total records: 347
  - Number of features(columns): 11 features

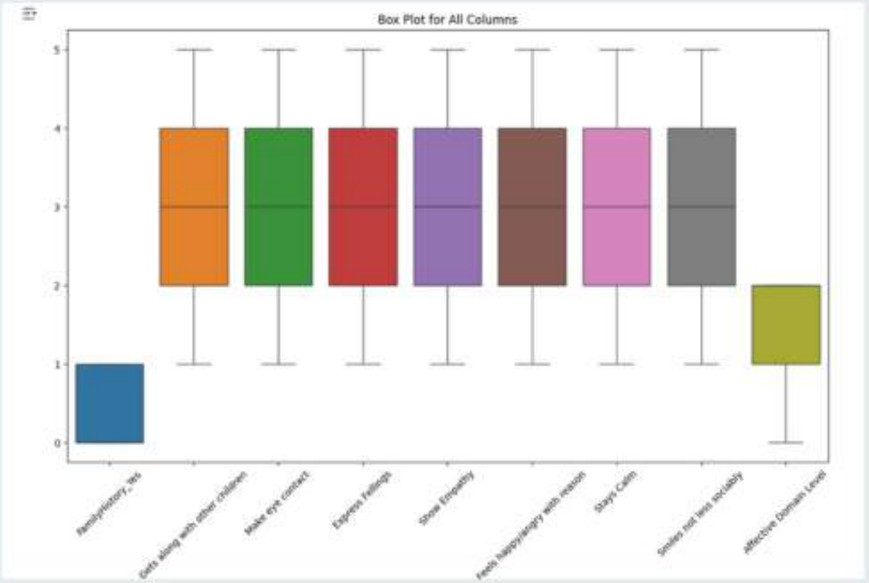
	A	B	C	D	E	F	G	H	I	J	K
1	Gender	Age	Family Member w/ Autism	Gets along with other children	Make eye contact	Express Fellings	Show Empathy	Feels happy/angry with reason	Stays Calm	Smiles not less sociably	Affective Domain Level
2	Female	14	Yes	3	3	4	2	3	2	2	Severe
3	Male	5	No	2	3	3	2	4	2	4	Severe
4	Female	12	No	5	4	5	5	4	3	5	Mild
5	Female	11	Yes	4	5	5	4	3		4	Moderate
6	Female	8	No	3	3	3	3	5	3	3	Moderate
7	Female	10	No	4	5	5	3	3	2	1	Moderate
8	Male	5	No	5	2	5	2	5	1	5	Moderate
9	Male	6	No	4	4	3	4	3	3	5	Moderate
10	Male	6	No	2	2	2	1	2	1	4	Severe
11	Male	10	No	4	4	3	3	3	4	5	Moderate
12	Female	10	No	3	2	4	2	3	2	2	Severe
13	Female	6	No	4	4	4	4	4	3	4	Moderate

Data Analysis


```
categorical_cols = ['Gender', 'Family Member w/ Autism']

#one-hot encoded columns
encoded_features = pd.get_dummies(datasetAD[categorical_cols], columns=categorical_cols, prefix=['Gender', 'FamilyHistory'], drop_first=True)
datasetAD = datasetAD.drop(columns=categorical_cols)
datasetAD = pd.concat([datasetAD, encoded_features], axis=1)

#converted to integers
bool_columns = datasetAD.select_dtypes(include=['bool']).columns
datasetAD[bool_columns] = datasetAD[bool_columns].astype(int)
```

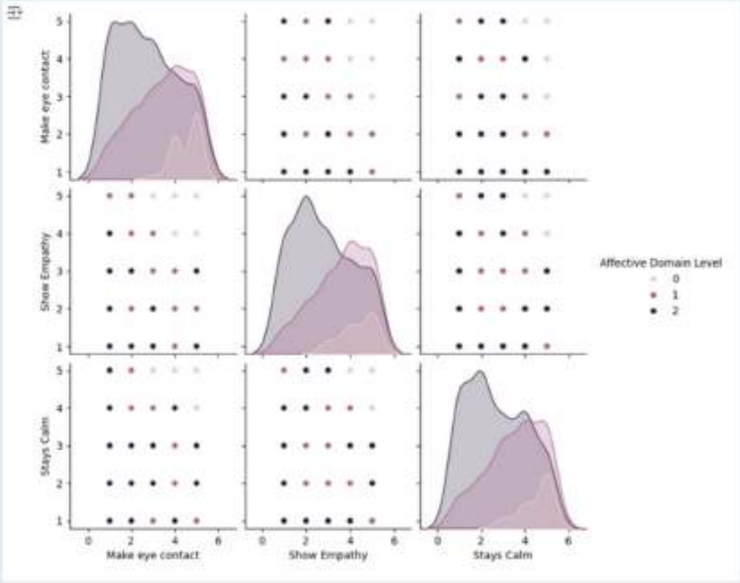
```
[ ] #encode 'Affective Domain Level' attribute

    ADLevel_mapping = {'Severe': 2, 'Moderate': 1, 'Mild':0}

    datasetAD['Affective Domain Level'] = datasetAD['Affective Domain Level'].map(ADLevel_mapping)
    print(datasetAD["Affective Domain Level"].unique())
```

```
[2 0 1]
```




```
#check null values
datasetAD.isnull().sum().sort_values(ascending=False)
```

	0
Show Empathy	2
Stays Calm	2
Make eye contact	1
Gender	0
Age	0
Family Member w/ Autism	0
Gets along with other children	0
Express Fellings	0
Feels happy/angry with reason	0
Smiles not less sociably	0
Affective Domain Level	0



Model Building

- Training set assigned 70%, validation set assigned 15%, and remain data assigned testing set.

- Classification models developed:
  - Random Forest
  - K-Nearest Neighbor (KNN)
  - Naive Bayes
  - Logistic Regression
  - Neural Network

- Based on the results from these methods, compare accuracy, precision, recall, and F1-score to decide the best model for the given dataset.

	A	B	C	D
1	**Model**	**Validation Accuracy**	**Test Accuracy**	**Reason**
2	Random Forest	85%	72%	Performance improved post-hypertuning, but recall is still lower on the test set.
3	KNN	63%	68%	Struggles with recall on the validation and test sets.
4	Naive Bayes	92%	79%	Consistently high performance on training and validation sets, but test set recall is lower.
5	Logistic Regression	85%	91%	**BEST MODEL [Consistently achieved the highest accuracy and balanced precision, recall, and f1-scores.]**
6	Neural Network	88%	81%	Struggles with recall, though performs well in training set accuracy.
7				

Model Building

- Create a function to predict each child's "Affective Domain Level" based on the emotional and social behavior features given to the user. It will classify the input into one of the following pre-defined categories:

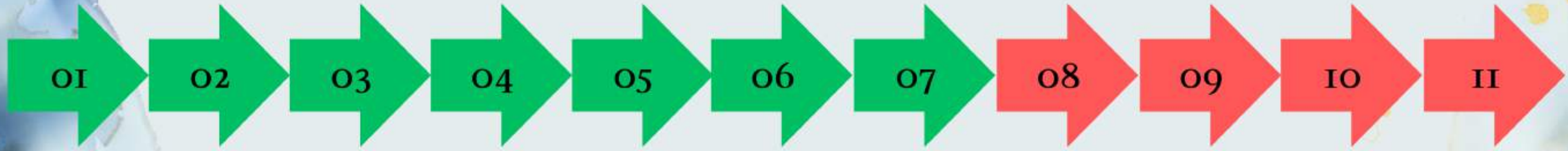
  Mild -> 0

  Moderate -> 1

  Severe -> 2

- The approach ensures customized recommendations are provided based on each child's affective domain needs.

Project completion plan



01. Background Search
02. Data Gathering
03. Data Analysis & Pre-processing
04. Model Building and Validation
05. Best Model Selection
06. Affective Domain Level Prediction
07. Customized Recommendations
08. Progress Tracking
09. Building a web app & integrating the models
10. Testing
11. Releasing to the required institutes

Completed 60%

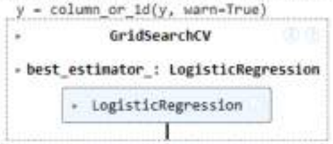
To do 40%

Proof Of Work

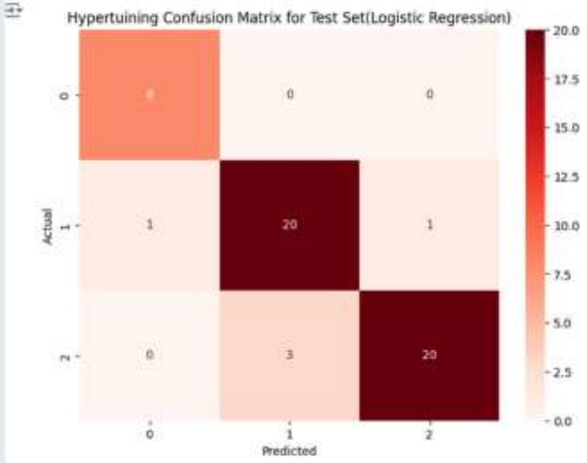
```
[ ] param_grid = {
        'C': [0.001, 0.01, 0.1, 1, 10, 100],
        'penalty': ['l2'],
        'solver': ['lbfgs', 'saga'],
        'max_iter': [1000, 2000]
    }

    #initialize the logistic regression model
    log_reg = LogisticRegression()

    # Use Stratified K-Fold cross-validation to ensure balanced class splits
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

    # Initialize GridSearchCV with stratified splits
    grid_search_log_reg = GridSearchCV(
        estimator=log_reg,
        param_grid=param_grid,
        cv=cv,
        scoring='neg_mean_squared_error',  # Use negative MSE as the scoring metric
        verbose=2,
        n_jobs=-1
    )
```

```
[ ] # Fit the grid search to the training data
    grid_search_log_reg.fit(X_trainB, y_trainB)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWar
  y = column_or_1d(y, warn=True)
```




```
[ ] cm = confusion_matrix(y_test, y_Testpred_best_log_regA)

    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Reds", xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title("Hypertuning Confusion Matrix for Test Set(Logistic Regression)")
    plt.show()
```




```
[ ] #classification report
    print("\nClassification Report for Training Set(Logistic Regression - Before Hypertuning):")
    print(classification_report(y_trainB, y_train_pred_log_regB))

    print("\nClassification Report for Validation Set(Logistic Regression - Before Hypertuning):")
    print(classification_report(y_val, y_val_pred_log_regB))

    print("\nClassification Report for Test Set(Logistic Regression - Before Hypertuning):")
    print(classification_report(y_test, y_test_pred_log_regB))
```

```
Classification Report for Training Set(Logistic Regression - Before Hypertuning):
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       252
           1       0.93      0.92      0.92       272
           2       0.94      0.93      0.94       276

    accuracy                           0.95       800
   macro avg       0.95      0.95      0.95       800
weighted avg       0.95      0.95      0.95       800


Classification Report for Validation Set(Logistic Regression - Before Hypertuning):
              precision    recall  f1-score   support

           0       0.40      1.00      0.57         2
           1       0.74      0.82      0.78        17
           2       1.00      0.85      0.92        33

    accuracy                           0.85        52
   macro avg       0.71      0.89      0.76        52
weighted avg       0.89      0.85      0.86        52


Classification Report for Test Set(Logistic Regression - Before Hypertuning):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       0.88      0.95      0.91        22
           2       0.95      0.87      0.91        23

    accuracy                           0.92        53
   macro avg       0.94      0.94      0.94        53
weighted avg       0.93      0.92      0.92        53
```

```
[ ] #get the best parameters
    best_params_log_regA = grid_search_log_reg.best_params_
    print("Best Hyperparameters for Logistic Regression:", best_params_log_regA)
```

```
Best Hyperparameters for Logistic Regression: {'C': 10, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
```

Proof Of Work

Get User Data

```
[ ]  def get_valid_ADinput(ADprompt):
        while True:
            user_ADinput = input(ADprompt).strip()
            if not user_ADinput:
                print("Please Enter a Score.")
            elif user_ADinput.isdigit() and 1 <= int(user_ADinput) <= 5:
                return int(user_ADinput)
            else:
                print("Invalid Score. Please Enter a Score between 1 and 5.")
```

Activities of Affective Domain Level

```
[ ]  def suggest_ADactivities(AD_Level):
        ADactivities = {
            "Mild": [
                "Emotion Identification with Facial Expressions",
                "Role-playing Social Scenarios",
                "Mindfulness Activities",
                "Social Stories",
                "Group Activities"
            ],
            "Moderate": [
                "Emotion Wheel",
                "Therapeutic Play",
                "Feelings Journal",
                "Social Skills Groups",
                "Guided Emotional Role-play",
                "Simple Breathing Techniques"
            ],
            "Severe": [
                "Picture Exchange Communication System (PECS)",
                "Sensory Activities",
                "Simple Emotion Identification",
                "Interactive Music or Sound Therapy",
                "Story Time with Emotions",
                "Calming Zones"
            ]
        }

        print(f"\nRelevant Activities for {AD_Level} Level:")
        activities = ADactivities[AD_Level]
        for ADactivity in activities:
            print(f"- {ADactivity}")
        print()
        print()

        return activities
```

```
def predictUserInput_affective_domain_level():
    print("Please enter the following details:")

    gender = input("Gender (Male=1, Female=0): ").strip()
    while gender not in ["0", "1"]:
        print("Invalid Gender. Please enter 1 for Male or 0 for Female.")
        gender = input("Gender (Male=1, Female=0): ").strip()

    age = input("Age: ").strip()
    while not age.isdigit() or int(age) <= 0:
        print("Invalid Age. Please Enter a valid age.")
        age = input("Age: ").strip()

    family_history = input("Family Member with Autism? (Yes=1, No=0): ").strip()
    while family_history not in ["0", "1"]:
        print("Invalid Status. Please enter 1 for Yes or 0 for No.")
        family_history = input("Family Member with Autism? (Yes=1, No=0): ").strip()

    print("\n\nPlease enter the marks for the following questions (1 to 5):")
    gets_along = get_valid_ADinput("Gets along with other children: ")
    eye_contact = get_valid_ADinput("Make eye contact: ")
    express_feelings = get_valid_ADinput("Express Feelings: ")
    show_empathy = get_valid_ADinput("Show Empathy: ")
    feels_reason = get_valid_ADinput("Feels happy/angry with reason: ")
    stays_calm = get_valid_ADinput("Stays Calm: ")
    smiles = get_valid_ADinput("Smiles not less sociably: ")

    # Convert inputs to a NumPy array
    user_data_UserInput = np.array([[
                                    int(family_history),
                                    gets_along,
                                    eye_contact,
                                    express_feelings,
                                    show_empathy,
                                    feels_reason,
                                    stays_calm,
                                    smiles]])

    # Predict the level using the logistic regression model
    predicted_ADlevel_UserInput = best_log_reg_model.predict(user_data_UserInput)

    # Map prediction to a readable level
    engagement_mapping = {2: "Severe", 1: "Moderate", 0: "Mild"}
    predicted_ADlevel = engagement_mapping[predicted_ADlevel_UserInput[0]]
    print("\nPredicted Affective Domain Level:", predicted_ADlevel)

    return suggest_ADactivities(predicted_ADlevel), predicted_ADlevel
```

```
predictUserInput_affective_domain_level()
```

```
Please enter the following details:
Gender (Male=1, Female=0): 1
Age: 10
Family Member with Autism? (Yes=1, No=0): 0


Please enter the marks for the following questions (1 to 5):
Gets along with other children: 22
Invalid Score. Please Enter a Score between 1 and 5.
Gets along with other children:
Please Enter a Score.
Gets along with other children: 4
Make eye contact: 1
Express Feelings: 1
Show Empathy: 1
Feels happy/angry with reason: 1
Stays Calm: 4
Smiles not less sociably: 5

Predicted Affective Domain Level: Severe

Relevant Activities for Severe Level:
- Picture Exchange Communication System (PECS)
- Sensory Activities
- Simple Emotion Identification
- Interactive Music or Sound Therapy
- Story Time with Emotions
- Calming Zones
```

Psychomtor Domain

Pehesarani W.K.A

Introduction

Psychomotor skills involve the coordinated function of the brain and muscles, enabling movements and physical activities.

Gross Motor Skills

Walking
Running
Jumping
Swimming
Climbing
Throwing and Catching

Fine Motor Skills

Writing
Drawing
Coloring
Buttoning Clothes
Typing

Background

- Target Problem: Autistic children often experience difficulties in enhancing their psychomotor skills, such as fine and gross motor abilities. Existing methods lack the ability to assess psychomotor levels and provide recommendations suitable for their developmental stage.

- Proposed Solution: The proposed system predicts the psychomotor level using ensemble methods and generates customized recommendations based on the child's level and age. With the support of parents, teachers, or peers, children can complete these tasks, and their progress is tracked to refine future recommendations.

Dataset

- Features:
  - Demographic Features: Gender, Age, Family history of ASD
  - Psychomotor Skills: Balance and stability, Grip strength, Large mussels coordination, Hand-eye coordination, Object manipulation, Utensil use, and clothing independence.
- Target variable:
  - The target variable is the psychomotor skill level, categorized as Mild, Moderate, or Severe
- Data volume:
  - Total records: 347
  - Number of features/columns: 11 features

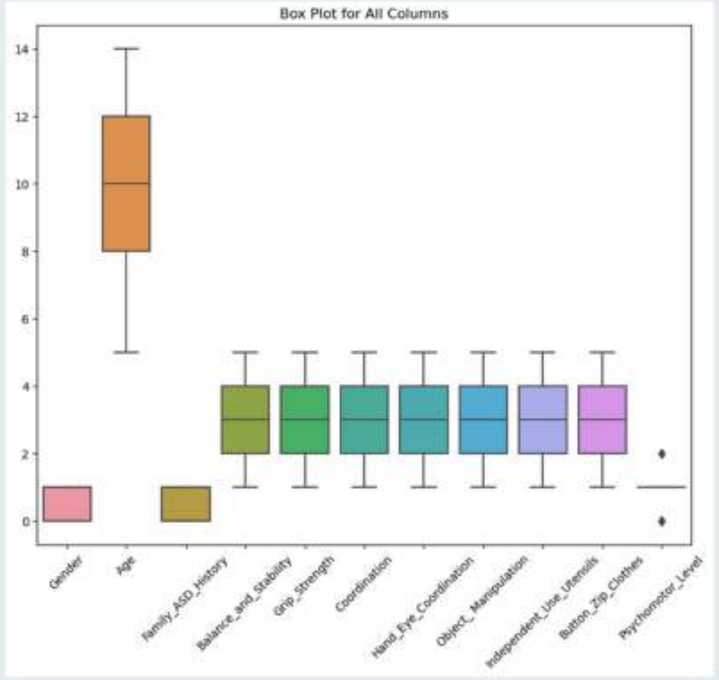
	A	B	C	D	E	F	G	H	I	J	K
1	**Gender**	**Age**	**Family_ASD_History**	**Balance_and_Stabilit**	**Grip_Strength**	**Coordination**	**Hand_Eye_Coordinati**	**Object_Manipulation**	**Independent_Use_Ut**	**Button_Zip_Clothes**	**Psychomotor_Level**
2	Female	8	No	Rarely	Maybe	Often	Rarely	Often	Maybe	Never Have	Moderate
3	Female	14	Yes	Always	Often	Rarely	Maybe	Maybe	Maybe	Never Have	Moderate
4	Female	10	No	Often	Maybe	Never Have	Rarely	Never Have	Never Have	Maybe	Severe
5	Male	14	No	Always	Never Have	Maybe	Often	Maybe	Often	Never Have	Moderate
6	Male	12	No	Always	Never Have	Rarely	Never Have	Rarely	Never Have	Never Have	Severe
7	Male	14	Yes	Maybe	Never Have	Often	Maybe	Often	Maybe	Rarely	Moderate
8	Male	13	No	Always	Rarely	Often	Often	Always	Maybe	Often	Mild
9	Male	12	Yes	Never Have	Maybe	Often	Never Have	Rarely	Maybe	Always	Moderate
10	Male	7	Yes	Often	Often	Often	Often	Maybe	Often	Always	Mild
11	Female	12	No	Always	Rarely	Always	Rarely	Never Have	Maybe	Often	Moderate
12	Male	6	Yes	Rarely	Always	Rarely	Never Have	Maybe	Rarely	Often	Moderate
13	Male	6	No	Always	Rarely	Maybe	Never Have	Rarely	Rarely	Rarely	Moderate
14	Female	13	Yes	Maybe	Rarely	Always	Rarely	Rarely	Never Have	Often	Moderate
15	Male	8	Yes	Always	Often	Often	Rarely	Always	Often	Often	Mild

Data Analysis

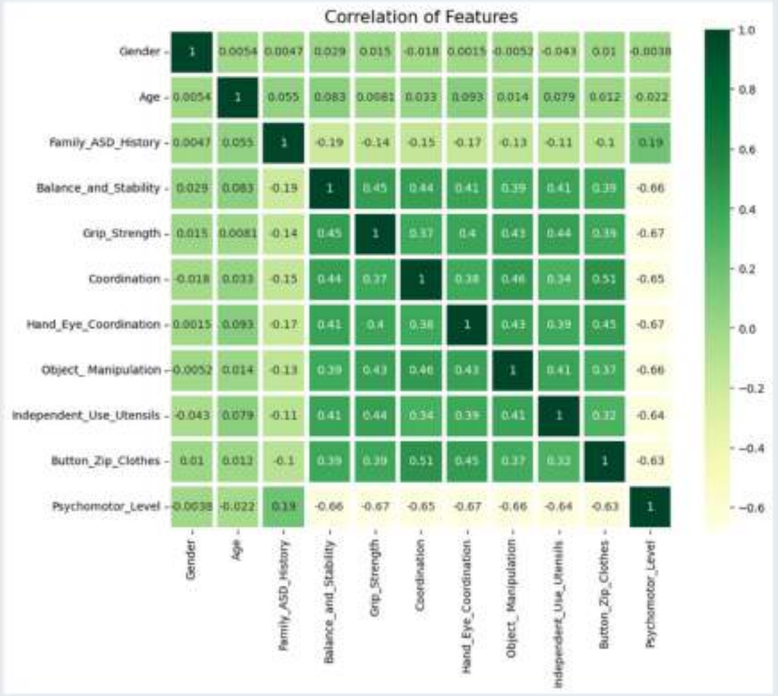
Mapped categorical variables

```
mappings = {
    "Gender": {"Male": 1, "Female": 0},
    "Family_ASD_History": {"Yes": 1, "No": 0},
    "Balance_and_Stability": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Grip_Strength": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Coordination": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Hand_Eye_Coordination": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Object_ Manipulation": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Independent_Use_Utensils": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Button_Zip_Clothes": {"Never Have": 1, "Rarely": 2, "Maybe": 3, "Often": 4, "Always": 5},
    "Psychomotor_Level": {"Mild": 0, "Moderate": 1, "Severe": 2}
}

# Map features to numerical values
for column, mapping in mappings.items():
    if column in df.columns:
        df[column] = df[column].map(mapping)

df
```

Outliers



Correlation matrix

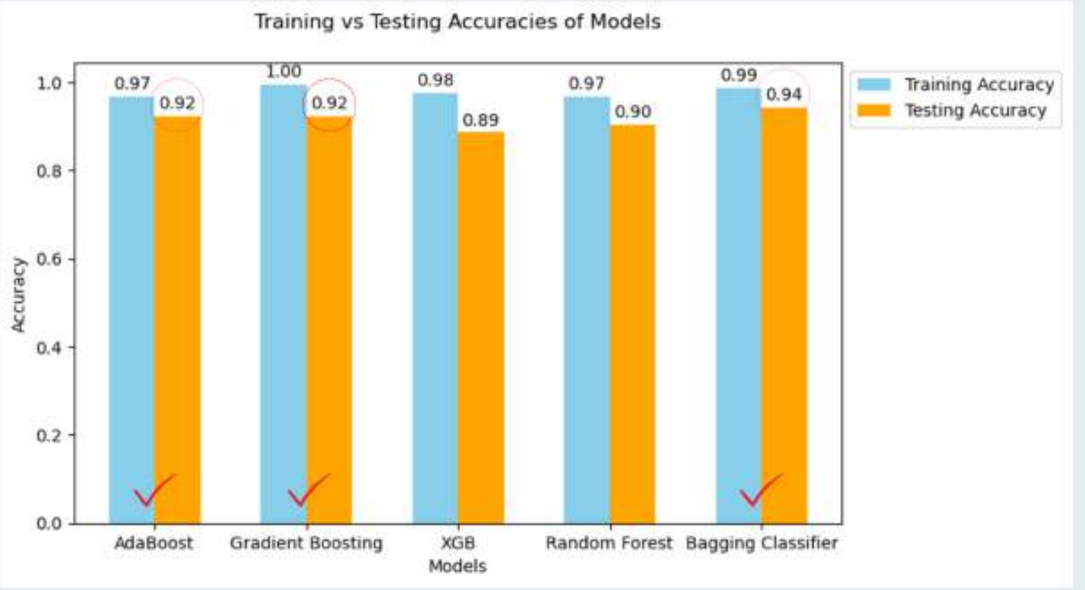


Model Building

1. Algorithm Selection
    - Used ensemble methods for better accuracy and reliability.
    - Bagging Algorithms: Bagging classifier, Random Forest
    - Boosting Algorithms: AdaBoost, GBM, XGBM

2. Model Training and Validation
    - Split data into training and testing sets.
    - Optimized model parameters using cross-validation to prevent overfitting.

3. Evaluation Metrics
    - Evaluated models performance using metrics like accuracy, precision, recall, and F1-score.
    - Chose the best-performing models based on predictive accuracy.



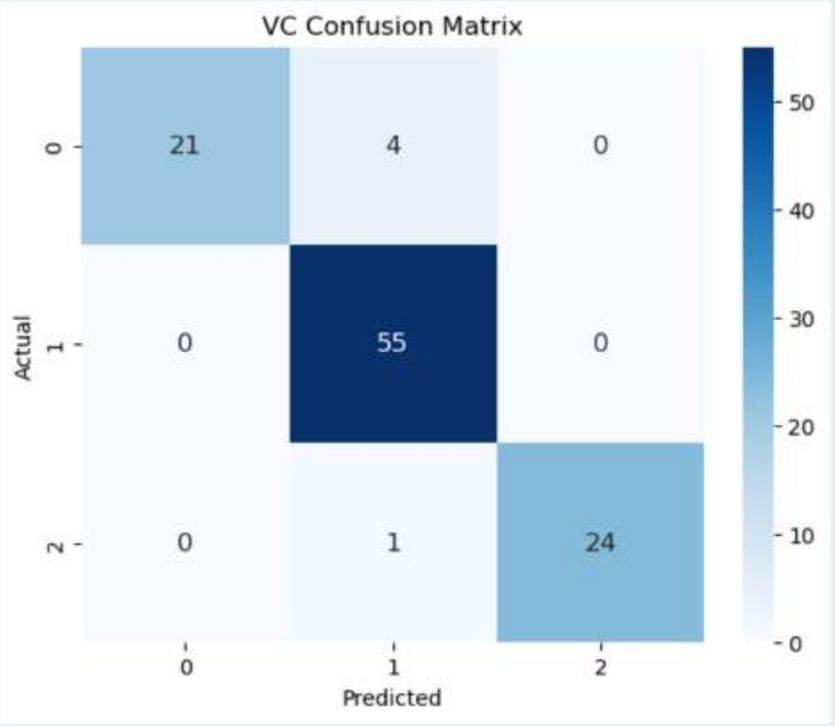
Model Building

4. Model Aggregation
   - Used a Voting Classifier to combine the predictions of the best-performing models (Bagging and Boosting algorithms) to achieve improved accuracy and robustness.
   - The ensemble approach aggregated the strengths of individual models, resulting in more reliable and consistent predictions.

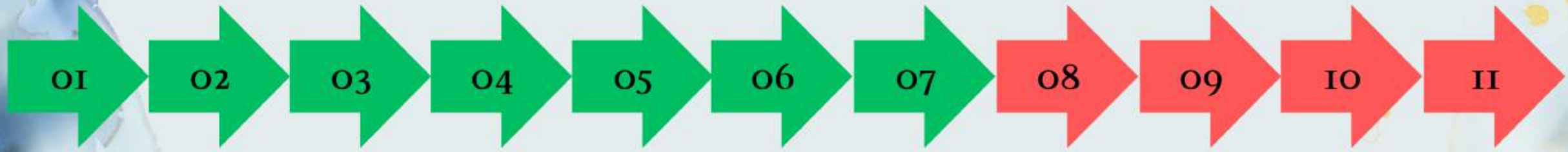
```
# Create a VotingClassifier with the best hyperparameters for voting='hard'
ensemble_classifier = VotingClassifier(
    estimators=[('gb', optimized_gb_model), ('ab', optimized_ada_model), ('bag', optimized_bagging_model)],
    voting='hard'  # Uses majority rule voting, where the class with the most votes is selected
)
```

✓ 0.0s

```
Training Accuracy: 0.987603
Test Accuracy: 0.952381
```



Project completion



01. Background Search
02. Data Gathering
03. Data Analysis & Pre-processing
04. Model Building and Validation
05. Best Model Selection
06. Psychomotor Domain Level Prediction
07. Customized Recommendations
08. Progress Tracking
09. Building a web app & integrating the models
10. Testing
11. Releasing to the required institutes

Completed 60%

To do 40%

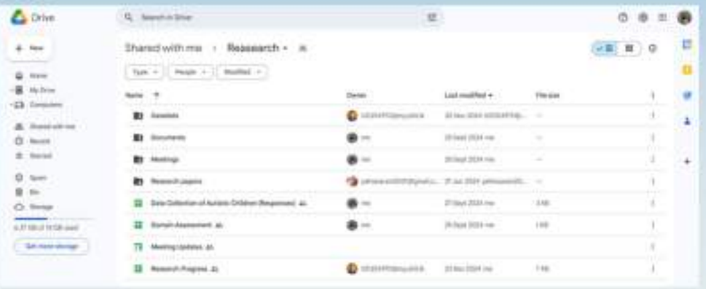
Standards and Knowledge Utilization

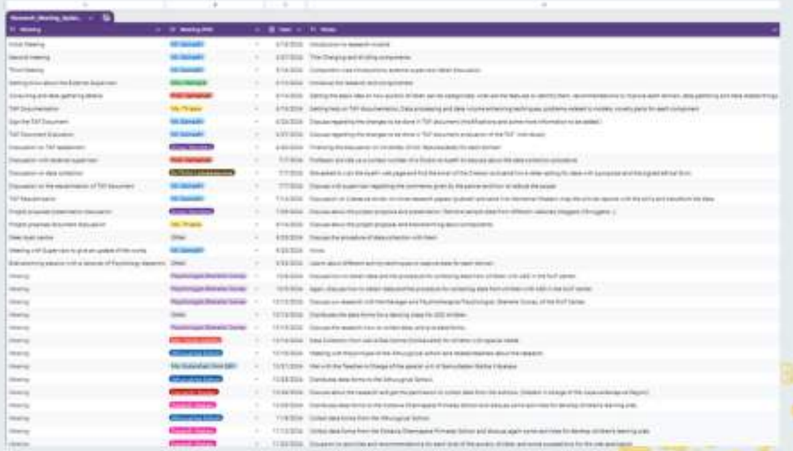
- Key pillars of data science and IT utilized in the implementation:
  - Data Management
  - Usage of Classification models, Clustering techniques and Ensemble methods
  - Progress Tracking Functionality

- Technologies: Python, Google Collab, Scikit-learn, and GitHub.



- Best Practices:
  - Followed coding standards (indentations, commenting, use of functions)
  - Efficient version control (GitHub)
  - Managing a log to track progress
  - Using cloud based platform to store related documents, images and references.





- For risk mitigation:
  - Expert validation on recommendations.
  - Feedback from educators and parents.
  - Weekly reviews and discussions with the team for consistent progress.

Commercialization

- Release pilot version to data-providing centers.
- Collect feedback from these centers.
- Develop the application further with advanced features based on feedback.
- Release the improved application to the public.

System Design and Implementation

Web Application

- Tailored for parents, peers, and educators to access recommendations and progress reports.

Design Approach

- Minimal and user-friendly design for easy accessibility.

Development Methodology

- React-based web application.
- Hosted on Microsoft Azure platform for reliable deployment and scalability.

Integration

- Backend support for machine learning based techniques and progress-tracking functionality.

Q & A

THANK YOU!