How do Parenting Methods Impact Child Development

1st Klarissa Navarro
Department of Engg and Comp. Science
California State University Fullerton
Fullerton, USA
klar@csu.fullerton.edu

2nd Kanika Sood
Department of Engg and Comp. Science
California State University Fullerton
Fullerton, USA
kasood@fullerton.edu

3rd Amal Suresh
Department of Engg and Comp. Science
California State University Fullerton
Fullerton, USA
amalsuresh@csu.fullerton.edu

***Abstract*—Child development is the process of a child developing the foundation of their health, learning capabilities, and behavioral characteristics. Each individual undergoes this development as a child, and it's essential to understand how parental parenting habits impact their child. Parents and children from primary to secondary schools were part of a study that answered questionnaires pertaining to parenting styles and their children's habits and characteristics. We use different classification techniques, such as KNN, Random Forest, SVM, and AdaBoost to create a predictive model to assign different areas of child development with a ranking—from 1 (high) to 4 (low). Our results demonstrate that the best performance is obtained from Random Forest, a child's development can be predicted based on parenting habits.**

***Index Terms*—child development, parenting habits, parenting style, child characteristics, KNN, SVM, RF, AdaBoost**

I. Introduction

Parenting is a complex and evolving process, with no single answer to the question, "How to be a good parent?" Throughout history, many researchers have explored how various parenting behaviors and actions influence children's development. While the methods and techniques for effective parenting continue to change, advancements in machine learning offer a new way to understand these influences. By analyzing patterns in parenting behaviors, machine learning can help identify how different parenting habits shape child development. Specifically, classification—a supervised modeling approach—can predict a child's ranking in various developmental areas.

Child development encompasses several domains, including physical, cognitive, communicative, and social/emotional skills. From these general domains, the child characteristics that will be used for this study will focus primarily on a child's level of attentiveness (P7ATTENI), behavior (P7BEHAVE), learning and thinking skills (P7SOLVE), and independence (P6SAMEAG). Machine learning can help determine where a child stands in relation to others in these key areas, providing valuable insights into the potential impacts of parenting techniques. For instance, in the context of attentiveness, a (1) would mean "better than other children their age," (2) is "as well as other children," (3) is "slightly less well than other children," and (4) is "much less well than other children."

To address this problem, we employ several classification techniques, including the classification techniques K-Nearest Neighbor (KNN), Support Vector Machines (SVM), Random Forest (RF), and Adaptive Boosting (AdaBoost). Using responses from questionnaires about parenting styles and children's habits and characteristics, we train these models to predict the correct class for each child's characteristics. We then evaluate the performance of these classifiers, comparing accuracy, precision, recall, and f-score to assess how well they capture the relationships between parenting behaviors and child development. Our results show that Random Forest outperformed the other classifiers regarding all evaluation metrics.

The rest of the paper is organized based on the following: Section 2 provides background information. Section 3 describes our dataset and data preprocessing steps. Section 4 explains our methodology. Section 5 presents and analyzes our results, and finally, Section 6 concludes with a summary and suggestions for future work.

II. Background

A. Supervised Learning

Supervised learning is a fundamental technique in machine learning where a model is trained on a labeled dataset. In this process, each data point is associated with one or more input features, along with a corresponding target label. The model learns to map these input features to the correct target labels during training. Once the model is trained, it can predict the target label for new, unseen data instances by applying the learned mapping from the training process.

B. Classification

Classification is a key method within supervised learning, which assigns a data point to one of several predefined classes based on its descriptive features. With multi-class classification, a target label consists of more than two classes, and the model will assign a data point to one of the classes. For example, all the target labels of our project are divided into four classes:

- class '1': above average
- class '2': average
- class '3': slightly below average
- class '4': substantially below average

To solve this problem, we employ non-linear classifiers to model non-linear relationships between the descriptive and target variables to establish non-linear decision boundaries to capture complex patterns. The following are used to solve the problem:

a) K-Nearest Neighbor (KNN): This classifier classifies a data point by identifying the 'k' nearest neighbors based on a chosen distance metric, such as Euclidean or Manhattan distance, and assigning the most frequent class label among those neighbors. For example, if k = 5 and we use a particular distance metric, the algorithm calculates the distance between the new data point (from the test set) and all points in the training set. It then selects the five data points with the smallest distances to the test point. For this problem, we classify the neighbors into one of the following categories: "above average," "average," "slightly below average," or "very below average." The most common label among the five will be assigned to the new data point. For instance, if three of the five neighbors are labeled "average," the new data point will be classified as "average." The choice of 'k' is crucial as it influences the classification outcome, and finding the optimal value of 'k' depends on the specific problem being addressed.

b) Support Vector Machines (SVM): This classifier separates classes in a dataset using a linear hyperplane, which serves as a decision boundary to categorize data points. The closest data points on either side of the hyperplane are called support vectors, and they play a critical role in determining the position of the hyperplane and the margin—the maximum distance between the support vectors and the hyperplane. A larger margin is preferable, as it provides a more apparent distinction between classes, reducing the risk of misclassification [1][2]. Most noticeably, a linear hyperplane works well for linearly separable problems, where a straight line can separate classes. However, in many real-world cases, such as the problem in this paper, the data may not be linearly separable. In such cases, it is essential to employ a kernel trick. This technique transforms the data into a higher-dimensional space, where the classes may become more easily separable. The specific kernel function used depends on the problem, and in our case, we used the Radial Basis Function (RBF) kernel. The RBF kernel computes the similarity between instances by measuring the Euclidean distance, making it practical for capturing complex, non-linear relationships between data points [3].

c) Decision Tree: This classifier is structured as a tree and is used to determine the class label of a data point. The root node represents the entire dataset, with each internal node corresponding to a decision based on a specific descriptive feature. The branches show the possible outcomes of these decisions, and the leaf nodes represent the predicted class labels. Starting at the root node, the algorithm recursively selects the best feature to split the data at each internal node, aiming to create subsets that are as pure as possible. This process is called information gain, which is calculated using entropy. Entropy measures the level of impurity/randomness in the dataset, specifically indicating how mixed the class labels are within the data. Decision trees strive to reduce entropy by creating splits that result in more homogeneous subsets (i.e., subsets with a single-class label). For example, a pure split occurs when a feature perfectly divides the data into distinct class labels. However, an impure split occurs if the resulting subsets contain mixed-class labels. The goal is to maximize information gain since a higher information gain indicates a more effective split that leads to greater purity in the resulting subsets [4].

d) Random Forest (RF): This classifier utilizes ensemble learning, a method that combines multiple models to improve the accuracy of class labeling. Specifically, it employs the bagging technique, a type of ensemble learning, where each model operates independently and has slight variations in its structure, ensuring that no models are identical. In the case of Random Forest, this is represented by 'n' decision trees, where 'n' is the total number of trees in the forest. Each decision tree casts a vote for the class label, and the majority vote among the trees determines the final class assignment for a new instance.

e) Adaptive Boosting (AdaBoost): This classifier utilizes boosting—a type of ensemble learning that creates one strong classifier from multiple weak learners, such as a decision stump (a decision tree with one split) or low-depth decision tree. It begins with training a weak classifier and identifying the proportion of misclassification. The next model will try to re-correct the error from the previous model, and this process continues (building a model that improves the performance from the prior model) until the maximum number of models is added or reaches the absolute correct classification [5].

III. Dataset and Preprocessing

This section describes our dataset and the process we took to perform data preprocessing.

A. Raw Dataset

According to [6] and [7], a study was conducted to collect reliable data pertaining to a child's development from multiple sources, including children, their families, teachers, schools, and care providers. The information collected involves the home environment, home educational activities, school environment, classroom environment, classroom curriculum, teacher qualifications, and before/after school care (i.e., factors that can influence child development). Surveys [6] were conducted from 1998-2007 for the same children from kindergarten to 8th grade, and then from 2010-2016 [7] with a new batch of students from kindergarten to fifth grade. There are 21,410 subjects in the 1998-2007 study and 18,175 in the 2010-16 study. Each dataset has more than 2,000 features; and when we combine them, we have more than 40,000 data points.

B. Data Preprocessing

Before combining the datasets, there are two steps to narrowing down the features:

- When downloading the data, extra information, such as IDs, is included by default. We believed it was necessary to delete the additional information because some ID

variables would exist in one dataset but not the other. Since IDs are unique and can't be replaced or generated, we couldn't keep them. Also, the paper's primary focus was to see a parent's influence on their child, so we only looked at features related to child development, home environment, and home educational activities. We grabbed a handful of descriptive features from each of these general categories to apply to our problem. As a result, we have 104 descriptive features and 18 potential target labels.
- Now, our first major issue is the naming conventions for both datasets. Each study has many similarities in the types of questions they ask and the answers, but didn't have the same names. For example, from the first dataset, the feature "P4READBO" represents the question, "Do you read books to your child?" The same question is in the second dataset but labeled "P4READBK." We manually review the feature labels and rename labels from the (second dataset) 2011 data to match those in the (first dataset) 1999 data.

With these two steps, we narrow the features and finally combine the datasets. Once we get the data together, some more actions were needed to improve the overall data to perform our machine learning techniques:

- Once combined, the next step is to deal with missing values. Since the study is done through questionnaires sent to parents and teachers, there are many questions that aren't answered. This led to significant gaps in our data. From the 104 descriptive features, rows with 57% or more missing values are dropped. Since all of our descriptive features are categorical, the most appropriate method for handling the missing values is to impute them with the mode. Consequently, some data points have a 0 or negative value assigned to a feature, which is equivalent to a missing value. To address this, we first deleted any feature where the mode was 0 or negative. Next, for the remaining descriptive features, we replaced all 0's or negative values with the mode of the corresponding feature.
- To find the correlation between categorical features, we use Chi-squared tests to narrow down the best descriptive features to use in our final dataset by running the test with each of our target features iteratively.
- We also perform one-hot encoding on the descriptive features for compatibility with the machine learning models we choose.
- Original, we had 18 potential target labels: P1SOLVE, P4SOLVE, P6SOLVE, P7SOLVE, P1BEHAVE, P4BEHAVE, P6BEHAVE, P7BEHAVE, P1ATTENTI, P4ATTENTI, P6ATTENTI, P7ATTENTI, P1ACTIVE, P4ACTIVE, P6ACTIVE, P7ACTIVE, P6SAMEAG, and P7SAMEAG. As you may have noticed, most labels look similar; the only difference is the number assigned to a feature. Each number assigned to these labels represents a different time during which the evaluation occurred during the study. We use cross-validation to reduce the original target label set to 4 target labels (P7SOLVE, P7BEHAVE, P7ATTENTI, and P6SAMEAG).
- We perform one-hot encoding on our data, it drastically increases the number of dimensions. To combat this, we employ a Multiple Correspondence Analysis (MCA). It helps pick out the top 20 features that explains the most variability in the data.
- The last step is to deal with the imbalanced classes within all our target labels. Hence, we use SMOTE, a resampling technique to create synthetic data points to balance the classes.

All these data preprocessing techniques finally allow us to perform KNN, Random Forest, SVM, and AdaBoost models.

IV. Methodology

In this section, we cover the methodology for this work. We go into detail about the features we select and the classifiers that are used on our data. For each target label, we test the full feature set and a reduced feature set with all the models.

The first classifier we use is was K-Nearest Neighbor, as it is very useful in classification problems. Since our data is categorical (nominal), the only thing we had to do to our dataset was to normalize it by one-hot encoding. For the K-Nearest Neighbor model, K-fold cross-validation was used to find the optimal number of neighbors, which was 5, for the model.

Using a Support Vector Machine was also a good option because of its efficiency with high-dimensional data. When the kernel parameter for the model was set to "rbf," it generalized well as the data followed a non-linear relationship. The only problem is that the time to run is very high, taking about 6 minutes per target feature for the full dataset.

Random Forest is one of the most powerful and versatile machine-learning algorithms and the improved version of Decision Trees. It's capable of handling large datasets, capturing complex, non-linear relationships, and performs with high accuracy. Since Random Forest incorporates the bagging technique from ensemble learning, this greatly factors why it has great accuracy because instead of listening to a single decision, you can see how multiple decision trees can lead to other decisions. Now, taking all the decisions versus one decision, improves the accuracy of what the likelihood of assigning the data point to the correct

AdaBoost is not very successful in providing good results. After setting the max depth of the base tree as 5 and

TABLE I: The class distribution for a child's attentiveness (P7ATTENI)

Class Distribution for P7ATTENI	
Class Label	No. of Instances in a Class
1	4,089
2	18,123
3	2,151
4	423

n_estimator as 100, it performed very poorly. It assigned a uniform weight for all training samples. Then, it created and ran a decision tree of max depth 5, identified misclassified samples, and assigned a weight to the tree based on the accuracy. It updated the weights of the training samples from the results of the decision tree, and repeated the process of creating more decision trees and adjusting weights for 100 iterations.

V. Result and Analysis

In this section, we divide the data into training and testing sets with an 80-20 split, then develop KNN, SVM, RF, and AdaBoost models for each target label: P7BEHAVE, P7SOLVE, P7ATTENTI, and P6SAMEAG. We present the results for each model and provide a detailed analysis of their performance.

A. Imbalanced Classes

Initially, when we assessed the model's performance using precision, recall, f1-score, and accuracy, we observed class imbalances in all target variables. This imbalance caused the models to be biased toward the majority class, resulting in poor generalization of new data. For instance, "Table I" displays the distribution of the target variable for child attentiveness (P7ATTENI), while "Table II" presents the evaluation scores for this variable, highlighting the impact of class imbalance.

There are various methods for evaluating model performance, and "Table II" outlines the metrics we used. The following explains how these metrics function and how class imbalance influences their scores:

- *Accuracy* measures the proportion of correct predictions out of all predictions, but it can be misleading in imbalanced datasets. A model could achieve high accuracy by correctly predicting the majority class most of the time.
- *Recall* measures the proportion of actual positive instances that are true positives (i.e., correctly identified).
- *Precision* measures the proportion of predicted positives that are true positives (i.e., focuses on the accuracy of positive predictions).
- *F1-score* is particularly useful for imbalanced classes since it combines precision and recall—providing a more reliable evaluation of model performance.

As shown in "Table III," RF consistently achieves the highest f1-score among the four models, with most target labels achieving around 0.90. In contrast, the other models exhibit significantly low performance for all targets, with AdaBoost performing the worst. For the other metrics, similar to the results in "Table II", the overall pattern for each target is consistent with which model performs better. Due to the class imbalance, the models that showed high values, such as KNN and RF were biased towards the majority class, leading to attractive results.

The poor performance for most of the models left us with two choices: undersampling or oversampling. Undersampling would mean eliminating data points from the majority classes and evening it out with the minority class. If we did that to all our target features, our dataset would've been too small to train and get proper results. As a result, we chose to pursue oversampling techniques. Oversampling adds data to the minority classes to balance it with that of the majority class.

TABLE II: The evaluation of P7ATTENI when the class was imbalanced

Score for Each Model of an Imbalanced P7ATTENTI				
Metric	KNN	RF	SVM	ADABOOST
Precision	0.62	0.93	0.93	0.27
Recall	0.86	0.85	0.25	0.26
F1-score	0.69	0.89	0.21	0.25
Accuracy	0.74	0.93	0.73	0.69

TABLE III: F1-Score for Each Model for Each Target Label with Imbalanced Classes

Target Label	**F1-Score**			
	KNN	***RF***	***SVM***	***AdaBoost***
P7ATTENTI	0.69	0.89	0.21	0.24
P7SOLVE	0.44	0.53	0.21	0.24
P7BEHAVE	0.67	0.90	0.21	0.23
P6SAMEAG	0.69	0.89	0.21	0.25

B. Balanced Classes

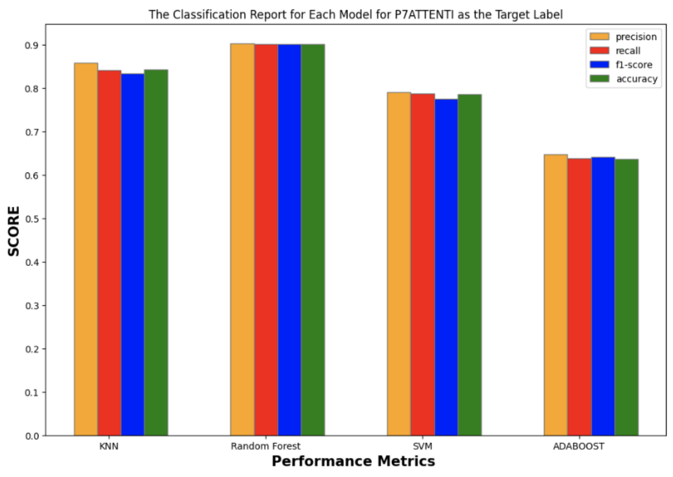
Due to the poor evaluation performance due to having imbalanced classes, we used SMOTE (Synthetic Minority Oversampling Technique). From a data point, this algorithm will randomly select one instance from its 'k' nearest neighbors of that data point. Then, it generates an artificial instance between the selected neighbor and the original data, creating a new, unique sample that's similar to the pre-existing cases of the minority class [8]. Overall, SMOTE helps balance the class distribution and improve the model's performance.

C. Learner Comparison

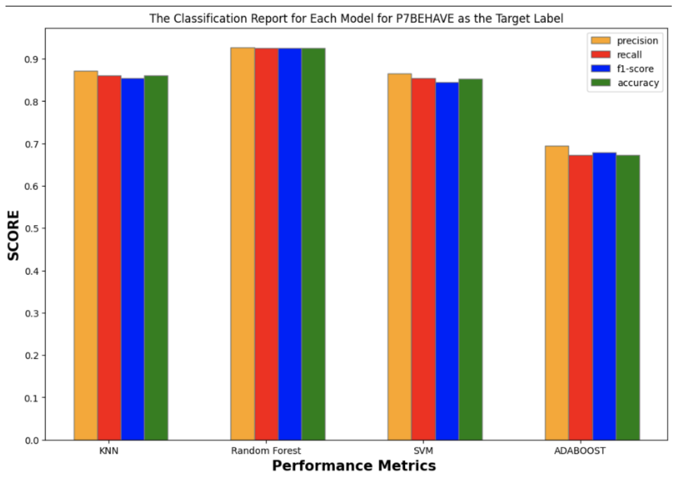
The classification report of the four classifiers can be compared and visualized by "Fig. 1". When precision, recall, f1-score, and accuracy are all compared, the RF model came out on top with the highest metrics across all target features. KNN comes a close second, followed by SVM, and finally, AdaBoost, performing the worst out of the bunch.

D. Analysis of Best Performance

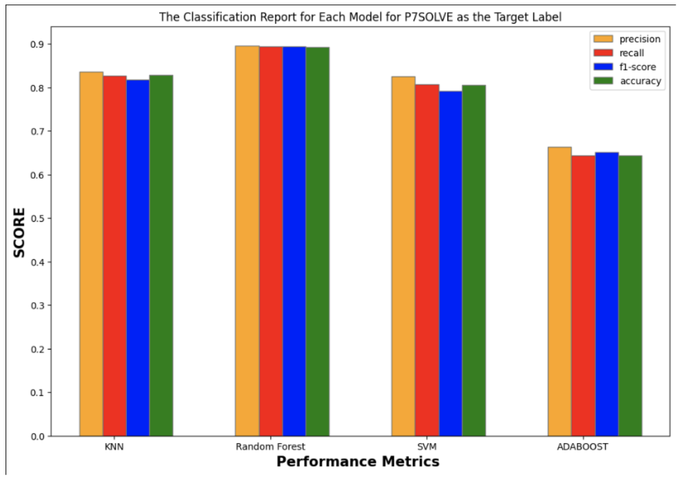
Our best-performing model is RF for every target label across the different evaluation metrics (recall, precision, f1-score, and accuracy) with scores ranging from 0.85 to 0.95 as shown in "Fig. 1a" to "Fig. 1d". Balancing the classes was critical for getting things working and achieve high performance metrics. This prevented the possibility of overfitting the majority class and once it was dealt with, running the model was straightforward. A key contributor to the efficiency of the model were the one-hot encoded input varaibles. This streamlined the evaluation process for the decision trees to predict the target feature's class.



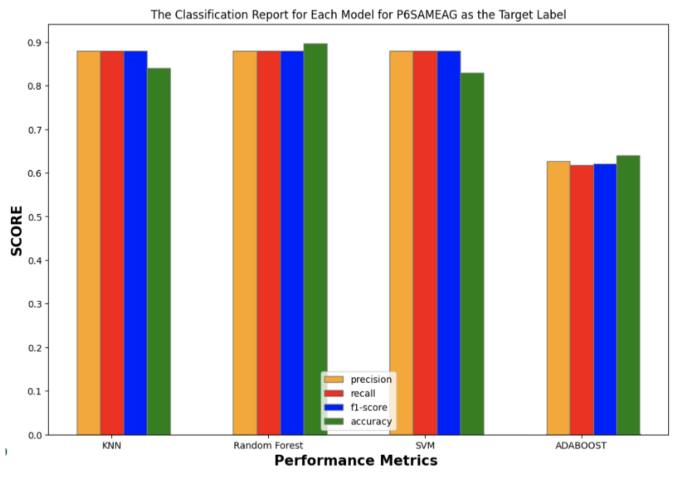
(a) Target Label: Attentiveness (P7ATTENTI)



(b) Target Label: Behavior (P7BEHAVE)



(c) Target Label: Critical Thinking (P7SOLVE)



(d) Target Label: Independence (P6SAMEAG)

Fig. 1: Performance Evaluation on KNN (left), RF (center-left), SVM (center-right), and AdaBoost (right) for each Target Label using precision (yellow), recall (red), f1-score (blue), and accuracy (green).

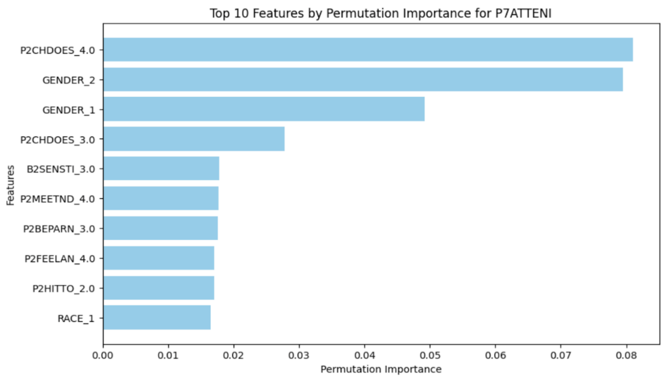
We also observed that using a balanced dataset allowed us to achieve a better balance between precision and recall, unlike when the dataset was imbalanced. With imbalanced data, it's common to see higher precision paired with lower recall, or higher recall with lower precision, as shown in "Table II". This is why the f1-score is so important—it provides a more comprehensive evaluation of how well precision and recall work together. Although the RF model performed well on the f1-score with the imbalanced data, it was likely overfitting to the majority class. In contrast, "Fig. 1" shows that with the balanced dataset, precision and recall are nearly equal, which is reflected in the improved f1-score.

E. Analysis of Worst Performance

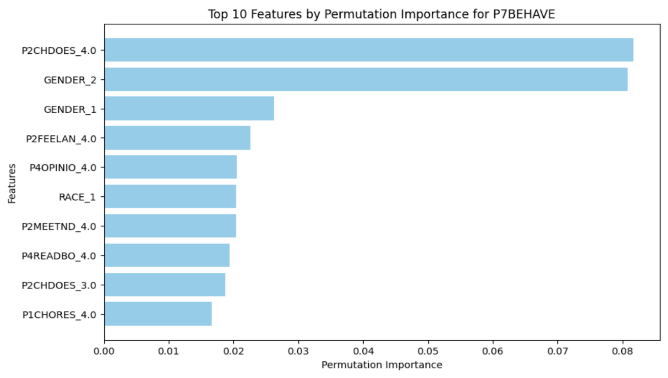
AdaBoost performed below par as it had the lowest performance compared to the other models as shown in "Fig. 1a to 1d". Our AdaBoost model uses a base tree of max depth 5 and 100 n_estimators. The main issue for the model was that our data was too complex for the weak learners due to the dimensionality. In comparison, the other classifiers delivered better performance, leading us to conclude that the AdaBoost model wasn't worth the computational cost.

F. Permutation Importance

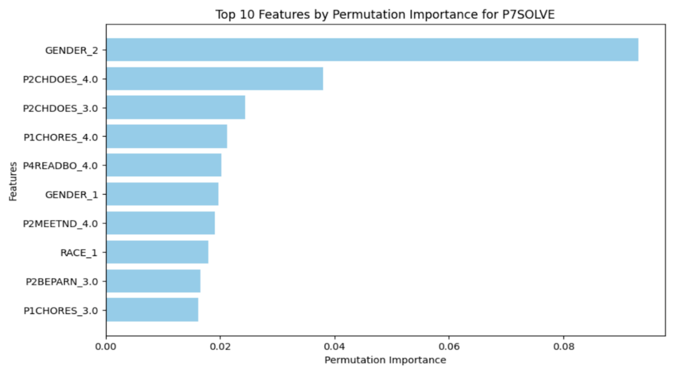
Since RF was the best-performing classifier, we used the permutation importance to help us identify what parenting habits had a greater impact on a target label, which is shown in "Fig. 2". Permutation importance is a method of classifying feature importance. According to [9], it first calculates all the performance metrics of the model on the validation set. Then, it takes one of the features and shuffles all its values. Finally, it recomputes the performance after the permutation of the feature and compares the difference in predictive quality. A positive value for permutation importance for a feature



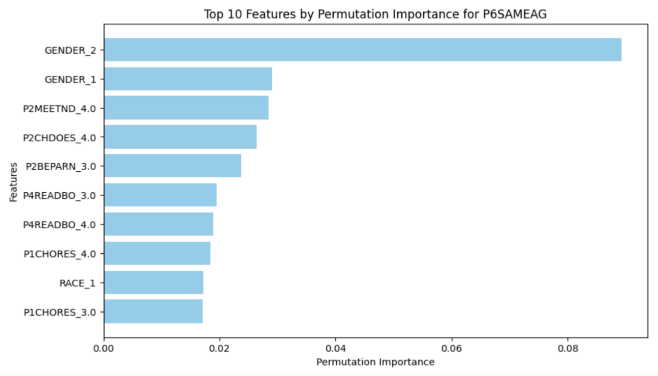
(a) Target Label: Attentiveness (P7ATTENTI)



(b) Target Label: Behavior (P7BEHAVE)



(c) Target Label: Critical Thinking (P7SOLVE)



(d) Target Label: Independence (P6SAMEAG)

Fig. 2: Top 10 Features by Permutation Importance for Each Target Label using Random Forest

indicates its importance in the model. For each of our target features, there were a few input variables in common in terms of importance. P2CHDOES (child does things that bother me), GENDER, P4READBO (I read books to my child), P1CHORES (how often does child do chores), and RACE all showed up multiple times for multiple target features.

VI. Conclusion and Future Works

From this work, we were able to evaluate, analyze, and compare four different classifiers. We compared the evaluation performance for each classification model using several parameters to predict the development of children based on parenting habits. Our results indicate that RF is the superior machine-learning algorithm for our problem. Using this classifier, due to its high performance across the several evaluation metrics we used, it provided us confidence that RF can produce a better accuracy of determining the descriptive features that have the most significant impact for each target feature (P7SOLVE, P7BEHAVE, P7ATTENTI, P6SAMEAG).

We believe there is still room for improvement in our project, with potential for scaling up. In future work, we plan to expand our dataset by incorporating data from the 2024 ECLS study once it becomes available [10]. Additionally, we will investigate which specific input features are most predictive of certain output classes. Our goal is to further reduce data dimensionality without compromising performance, or even enhancing it. We also plan to include descriptive features derived from teacher questionnaires about the school environment. Furthermore, the target label set should be broadened to encompass more general areas of child development. Above all, we are committed to making our data more inclusive by incorporating children with disabilities, diverse guardianship roles (e.g., siblings or grandparents), and family types (i.e., small/big family, adoptive, blood-related, etc.).

References

[1] "Support Vector Machine (SVM) Algorithm." geeksforgeeks. Accessed: Dec. 7, 2024. [Online.] Available: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[2] S. Saxena. "Beginner's Guide to Support Vector Machine(SVM)." analyticsvidhya.com. Accessed: Dec. 7, 2024. [Online.] Available: https://www.analyticsvidhya.com/blog/2021/03/beginners-guide-to-support-vectormachinesvm/#:~:text=A%20hyperplane%20is%20a%20decision,input%20features%20in%20the%20dataset.

[3] "Kernel Trick in Support Vector Classification." geeksforgeeks. Accessed: Dec. 7, 2024. [Online.] Available: https://www.geeksforgeeks.org/kernel-trick-in-support-vector-classification/

[4] O. Parmod. "Decision Trees." Medium. Accessed: Dec. 7, 2024. [Online.] Available: https://medium.com/@ompramod9921/decision-trees-6a3c05e9cb82

[5] ”Boosting in Machine Learning — Boosting and AdaBoost” geeksforgeeks. Accessed: Dec. 9, 2024. [Online.] Available: https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/

[6] “Early Childhood Longitudinal Program (ECLS) - Kindergarten Class of 1998-99 (ECLS-K),” nces.ed.gov. https://nces.ed.gov/ecls/kindergarten.asp

[7] “Early Childhood Longitudinal Studies Program (ECLS) - Kindergarten Class of 2010-11 (ECLS-K:2011),” Ed.gov, 2024. https://nces.ed.gov/ecls/kindergarten2011.asp

[8] J. Brownlee. ”SMOTE for Imbalanced Classification with Python.” Machine Learning Mastery. Accessed: Dec. 9, 2024. [Online]. Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[9] Scikit-learn, “4.2. Permutation Feature Importance,” *Scikit-learn User Guide*, 2024. [Online]. Available: https://scikit-learn.org/1.5/modules/permutation_importance.html

[10] “Early Childhood Longitudinal Studies Program (ECLS) - Kindergarten Class of 2023-24 (ECLS-K:2024),” Ed.gov, 2023. https://nces.ed.gov/ecls/kindergarten2024.asp