Analysis of the Florida iBudget Algorithm: Current Limitations and Proposed Quantitative Alternatives





Information Systems of Florida

September 5, 2025

**iBudget Algorithm Study**

Agency for Persons with Disabilities (ADP) Procurement Office: Tamara Harrington 4030 EWsplanade Way

Tallahassee, Florida 32399

Study prepared by Information Systems of Florida. Abbie David, PMP, PhD, Project Manager.

Jason Dillaberry, Client Partner

Matthew Fisher, JD, LLM, MBA, Principal Consultant Allison Kunerth, PhD, Senior Management Consultant Juan B. Guti´errez, PhD, Mathematician

Jessica Kemper, MPA, Senior Management Consultant Chris Klass, MPP Finance, Senior Management Consultant Annika Baeten, Management Consultant

Daniel Margolis, Associate Management Consultant [adavid@isf.com](mailto:adavid@isf.com)

Information Systems of Florida September 5, 2025

**Contents**

1. [Introduction](#_bookmark0) 6
   1. [Introduction](#_bookmark1) 7
   2. [Analysis of the Questionnaire for Situational Information (QSI): Data Types and](#_bookmark2) [Model Deficiencies](#_bookmark2) 8
      1. [QSI Data Structure and Question Categories](#_bookmark3) 8
         1. [Functional Status Questions (Q14-Q24)](#_bookmark4) 8
         2. [Behavioral Status Questions (Q25-Q30)](#_bookmark5) 9
         3. [Physical Status Questions (Q32-Q50)](#_bookmark6) 9
         4. [Composite and Additional Variables](#_bookmark7) 10
      2. [Structural Inconsistencies in the QSI Assessment Instrument](#_bookmark8) 10
2. [Previous Algorithm](#_bookmark9) 12
   1. [Previous Algorithm](#_bookmark10) 13
   2. [Statistical Methods Analysis](#_bookmark11) 13
      1. [Overview of Statistical Framework](#_bookmark12) 13
      2. [Multiple Linear Regression Foundation](#_bookmark13) 13
      3. [Statistical Assumptions and Limitations](#_bookmark14) 14
      4. [Box-Cox Power Transformation](#_bookmark15) 14
      5. [Model Selection via Bayesian Information Criterion](#_bookmark16) 14
      6. [Methodological Concerns](#_bookmark17) 15
         1. [Outlier Management](#_bookmark18) 15
         2. [Temporal Validity](#_bookmark19) 15
         3. [Person-Centered Alignment](#_bookmark20) 15
      7. [Implementation Implications](#_bookmark21) 15
   3. [Critical Deficiencies in Model 5b](#_bookmark22) 15
      * 1. [Counter-Intuitive Negative Coefficients](#_bookmark23) 16
        2. [Widespread Statistical Insignificance](#_bookmark24) 16
        3. [Violation of Distributional Assumptions](#_bookmark25) 16
        4. [Excessive Outlier Exclusion](#_bookmark26) 16
        5. [Limited Construct Validity](#_bookmark27) 16
        6. [Validation and Reliability Gaps](#_bookmark28) 16
   4. [Model 5b Implementation and Testing Framework](#_bookmark29) 17
      1. [Implementation Overview](#_bookmark30) 17
      2. [Program Execution](#_bookmark31) 17
         1. [System Requirements](#_bookmark32) 17
         2. [Execution Instructions](#_bookmark33) 17
         3. [Expected Output Structure](#_bookmark34) 17
      3. [Test Dataset Specification](#_bookmark35) 18
         1. [Dataset Structure](#_bookmark36) 18

|  |  |  |
| --- | --- | --- |
|  | [2.4.3.2 Test Case Coverage](#_bookmark37) . . . . . . . . . . . . . . . . . . . . . . . . . | 18 |
| [2.4.3.3 Variable Validation Framework](#_bookmark38) . . . . . . . . . . . . . . . . . . . | 19 |
| [2.4.3.4 Data Integrity and Realism](#_bookmark39) . . . . . . . . . . . . . . . . . . . . . | 19 |
| [2.4.4](#_bookmark40) | [Implementation Validation](#_bookmark40) . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
|  | [2.4.4.1 Computational Accuracy](#_bookmark41) . . . . . . . . . . . . . . . . . . . . . . | 20 |
|  | [2.4.4.2 Transparency and Traceability](#_bookmark42) . . . . . . . . . . . . . . . . . . . | 20 |
| [2.4.5](#_bookmark43) | [Testing Framework Applications](#_bookmark43) . . . . . . . . . . . . . . . . . . . . . . . | 20 |
| [2.4.6](#_bookmark44) | [Source code & Output](#_bookmark44) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |

1. [**Alternative Algorithms**](#_bookmark45) **32**
   1. [Alternative Algorithms](#_bookmark46) 33
      1. [Advanced Mathematical and Statistical Modeling Approaches](#_bookmark47) 33
         1. [Regularization Methods for High-Dimensional Data](#_bookmark48) 33
         2. [Sparse Estimation Techniques](#_bookmark49) 34
         3. [Robust Regression Approaches](#_bookmark50) 34
         4. [Machine Learning Approaches for Nonlinear Relationships](#_bookmark51) 35
         5. [Ordinal Regression Methods](#_bookmark52) 35
         6. [Hierarchical and Mixed-Effects Models](#_bookmark53) 35
         7. [Ensemble Methods](#_bookmark54) 36
      2. [Recommendations for Model Development](#_bookmark55) 36
   2. [Current Algorithm Analysis](#_bookmark56) 36
      1. [Mathematical Formulation](#_bookmark57) 36
      2. [Model Performance Metrics](#_bookmark58) 37
      3. [Critical Mathematical Limitations](#_bookmark59) 37
         1. [Outlier Dependency](#_bookmark60) 37
         2. [Temporal Validity Issues](#_bookmark61) 37
         3. [Transformation Bias](#_bookmark62) 37
   3. [Compliance Analysis with House Bill 1103](#_bookmark63) 38
      1. [Person-Centered Planning Deficiencies](#_bookmark64) 38
      2. [Data Currency Violations](#_bookmark65) 38
   4. [Proposed Alternative Algorithms](#_bookmark66) 38
      1. [Enhanced Linear Regression Approaches](#_bookmark67) 38
         1. [Algorithm A1: Robust Linear Regression](#_bookmark68) 38
         2. [Algorithm A2: Regularized Regression](#_bookmark69) 39
      2. [Machine Learning Ensemble Approaches](#_bookmark70) 40
         1. [Algorithm B1: Random Forest Regression](#_bookmark71) 40
         2. [Algorithm B2: Gradient Boosting with Custom Objective](#_bookmark72) 41
      3. [Hybrid Statistical-Clinical Approaches](#_bookmark73) 44
         1. [Algorithm C1: Two-Stage Hybrid Model](#_bookmark74) 44
         2. [Algorithm C2: Bayesian Hierarchical Model](#_bookmark75) 46
      4. [Person-Centered Optimization Approaches](#_bookmark76) 49
         1. [Algorithm D1: Multi-Objective Optimization](#_bookmark77) 49
         2. [Algorithm D2: Constrained Optimization with Fairness](#_bookmark78) 52
      5. [Modern Time-Aware Approaches](#_bookmark79) 56
         1. [Algorithm E1: Dynamic Regression with Time Effects](#_bookmark80) 56
         2. [Algorithm E2: Longitudinal Mixed-Effects Model](#_bookmark81) 60
      6. [Specialized Needs-Based Approaches](#_bookmark82) 64
         1. [Algorithm F1: Latent Class Mixture Model](#_bookmark83) 64
         2. [Algorithm F2: Support Vector Regression](#_bookmark84) 67
   5. [Implementation Framework and Validation](#_bookmark85) 70
      1. [Model Selection Criteria](#_bookmark86) 70
      2. [Validation Framework](#_bookmark87) 70
   6. [Recommendations and Implementation Roadmap](#_bookmark88) 72
      1. [Phased Implementation Approach](#_bookmark89) 72
   7. [Conclusion](#_bookmark90) 72

**Chapter 1**

**Introduction**

# Introduction

The Florida iBudget algorithm represents a critical component of the state’s developmental dis- ability services infrastructure, determining individual budget allocations for Home and Community- Based Services (HCBS) under the Developmental Disabilities Individual Budgeting waiver pro- gram. This system currently serves over 36,000 enrollees, making algorithmic decisions that directly impact the quality of life and service access for individuals with developmental disabil- ities across Florida. The algorithm’s role extends beyond mere budget calculation; it funda- mentally shapes how resources are distributed, what services individuals can access, and how person-centered planning principles are implemented in practice.

The enactment of House Bill 1103 in the 2025 legislative session has fundamentally altered the regulatory landscape for iBudget allocation methodologies. This legislation mandates a compre- hensive study to review, evaluate, and identify recommendations regarding the current algorithm, with particular emphasis on ensuring compliance with person-centered planning requirements under section 393.0662, Florida Statutes. The bill’s requirements extend beyond simple algorith- mic refinement, demanding a fundamental reassessment of how statistical methods align with person-centered planning principles and contemporary disability services philosophy.

This analysis addresses three interconnected questions that form the foundation for algorithm evaluation and redesign. First, we examine what the current algorithm accomplishes, including its mathematical formulation, variable selection, and operational mechanics. This examination reveals both the system’s statistical foundations and its practical implications for budget de- termination across diverse disability populations. Second, we identify critical weaknesses in the current approach, ranging from temporal validity issues stemming from outdated data to fun- damental limitations in capturing person-centered planning elements. These weaknesses extend beyond technical statistical concerns to encompass broader questions about algorithmic fairness, transparency, and compliance with evolving disability rights frameworks.

Third, we analyze specific areas where the current algorithm fails to meet the requirements established in House Bill 1103, particularly regarding person-centered planning integration, data currency, and algorithmic robustness. This compliance analysis reveals systematic gaps between the algorithm’s actuarial focus and the legislation’s emphasis on individualized, preference-driven service planning. The analysis demonstrates that addressing these compliance issues requires more than technical adjustments; it demands a fundamental reconceptualization of how algo- rithmic systems can support rather than constrain person-centered planning processes.

The analysis presented in this document extends beyond identifying weaknesses to propose systematic approaches for algorithmic improvement that address both technical limitations and compliance requirements. These approaches range from enhanced linear regression methods that maintain interpretability while improving robustness, to sophisticated machine learning techniques that can capture complex relationships between individual characteristics and sup- port needs, to hybrid approaches that combine statistical prediction with clinical judgment and person-centered planning elements.

The implementation strategy outlined in this analysis emphasizes phased deployment with comprehensive validation and monitoring to ensure that algorithmic improvements translate into meaningful improvements in service delivery and individual outcomes. This approach recognizes that algorithmic change in disability services carries profound implications for individual wellbe- ing and requires careful attention to unintended consequences and implementation challenges. This comprehensive analysis serves multiple audiences and purposes within Florida’s disability services ecosystem. For policymakers and legislative oversight bodies, it provides the technical foundation required by House Bill 1103 while translating complex statistical concepts into policy- relevant insights about algorithmic performance and compliance. For APD administrators and program managers, it offers practical guidance for algorithm selection and implementation while highlighting operational considerations that affect day-to-day service delivery.

For disability advocacy organizations and individuals receiving services, this analysis provides transparency about algorithmic decision-making processes and identifies specific areas where cur- rent methods may not adequately serve person-centered planning principles. For researchers and technical practitioners, it offers detailed methodological analysis and implementation guidance that can inform algorithm development and validation processes.

The analysis ultimately argues that effective algorithmic systems in disability services re- quire more than statistical sophistication; they demand explicit integration of person-centered planning principles, transparent decision-making processes, and ongoing adaptation to changing service delivery contexts. The current algorithm’s limitations stem not merely from technical deficiencies but from a fundamental misalignment between actuarial prediction methods and the individualized, preference-driven approaches that define quality disability services.

Moving forward, Florida’s iBudget system requires algorithmic approaches that can simul- taneously achieve statistical rigor, regulatory compliance, person-centered planning integration, and operational practicality. The alternative approaches presented in this analysis offer path- ways toward these multiple objectives while acknowledging the inherent tensions and tradeoffs involved in algorithmic design for disability services. The ultimate success of these approaches will depend not only on their technical implementation but on their ability to support rather than constrain the person-centered planning processes that remain central to effective disability services.

# Analysis of the Questionnaire for Situational Informa- tion (QSI): Data Types and Model Deficiencies

The Florida Questionnaire for Situational Information (QSI) Version 4.0 represents a compre- hensive assessment instrument designed to evaluate support needs for individuals with devel- opmental disabilities. This analysis examines the data structure, identifies critical deficiencies in the proposed statistical models, and recommends advanced modeling approaches to address these limitations.

## QSI Data Structure and Question Categories

The QSI contains comprehensive assessment data organized into three primary domains, each utilizing ordinal scales ranging from 0 (no support needed) to 4 (intensive support required).

### Functional Status Questions (Q14-Q24)

The functional status domain comprises 11 elements assessing daily living support needs:

* + - * + **Q14 - Vision**: Visual impairment assessment (0=no impairment, 4=constant assistance required)
        + **Q15 - Hearing**: Hearing impairment assessment (0=no impairment, 4=constant assis- tance required)
        + **Q16 - Eating**: Eating support needs (0=independent, 4=total assistance required)
        + **Q17 - Ambulation**: Mobility support needs (0=independent, 4=constant assistance re- quired)
        + **Q18 - Transfers**: Transfer support needs (0=independent, 4=total assistance required)
        + **Q19 - Toileting**: Toileting support needs (0=independent, 4=total assistance required)
        + **Q20 - Hygiene**: Personal hygiene support needs (0=independent, 4=total assistance required)
        + **Q21 - Dressing**: Dressing support needs (0=independent, 4=total assistance required)
        + **Q22 - Communications**: Communication support needs (0=no impairment, 4=constant assistance required)
        + **Q23 - Self-Protection**: Safety awareness and self-protection (0=independent, 4=constant supervision required)
        + **Q24 - Evacuation Ability**: Emergency evacuation capability (0=independent, 4=total assistance required)

### Behavioral Status Questions (Q25-Q30)

The behavioral domain encompasses 6 elements evaluating intervention needs for challenging behaviors:

* + - * + **Q25 - Self-Injurious Behavior**: Interventions for self-harm behaviors (0=none required, 4=physical/mechanical restraint used)
        + **Q26 - Aggressive/Hurtful to Others**: Interventions for aggressive behaviors (0=none required, 4=secure facility placement)
        + **Q27 - Destructive to Property**: Interventions for property damage (0=none required, 4=secure facility placement)
        + **Q28 - Inappropriate Sexual Behavior**: Interventions for sexual behavior issues (0=none required, 4=secure facility placement)
        + **Q29 - Running Away**: Interventions for elopement behaviors (0=none required, 4=se- cure facility placement)
        + **Q30 - Other Behaviors**: Other behaviors leading to separation (0=none required, 4=se- cure facility placement)

### Physical Status Questions (Q32-Q50)

The physical domain contains 19 elements addressing health and medical concerns:

* + - * + **Q32 - Self-Injury Related Injuries**: Injury severity from self-injurious behavior
        + **Q33 - Aggression Related Injuries**: Injury severity from aggressive behavior
        + **Q34 - Mechanical Restraints**: Use of protective equipment for behavioral issues
        + **Q35 - Emergency Chemical Restraint**: Use of emergency chemical interventions
        + **Q36 - Psychotropic Medications**: Psychotropic medication usage patterns
        + **Q37 - Gastrointestinal Conditions**: GI-related health issues including reflux, vomiting
        + **Q38 - Seizures**: Seizure-related conditions and management
        + **Q39 - Anti-Epileptic Medications**: Anti-seizure medication usage
        + **Q40 - Skin Breakdown**: Skin integrity issues
        + **Q41 - Bowel Function**: Bowel management needs
        + **Q42 - Nutrition**: Nutritional support requirements
        + **Q43 - Treatment (Physician Prescribed)**: Physician-prescribed treatments
        + **Q44 - Chronic Healthcare Needs**: Assistance with ongoing healthcare management
        + **Q45 - Individual’s Injuries**: Personal injury patterns
        + **Q46 - Falls**: Fall-related concerns
        + **Q47 - Physician Visits/Nursing Services**: Healthcare service utilization
        + **Q48 - Emergency Room Visits**: Emergency healthcare utilization
        + **Q49 - Hospital Admissions**: Inpatient healthcare utilization
        + **Q50 - Days Missed**: Activity missed due to illness

### Composite and Additional Variables

The QSI generates several composite scores and includes demographic variables:

* + - * + **FSum**: Functional status raw score (sum of Q14-Q24, range 0-44)
        + **BSum**: Behavioral status raw score (sum of Q25-Q30, range 0-24)
        + **PSum**: Physical status raw score (sum of Q32-Q50, range 0-76)
        + **Living Setting**: Six categorical levels ranging from family home to intensive residential care
        + **Age Groups**: Multiple categorical classifications (3-20, 21-30, 31+ years)

## Structural Inconsistencies in the QSI Assessment Instrument

The QSI exhibits several fundamental design inconsistencies that compromise its reliability as a standardized assessment tool. These include non-uniform scaling systems, unvalidated question exclusions, inconsistent temporal frameworks, and ad-hoc scoring rules that violate the instru- ment’s stated ordinal structure.

**Binary vs. Ordinal Scale Inconsistency (Q43)** Question 43 (Treatment/physician pre- scribed) employs a binary scale (0 or 4 only) while all other QSI questions utilize a consistent 5-point ordinal scale (0-4). The standard QSI scaling pattern follows: 0 = none, 1 = minimal, 2 = moderate, 3 = frequent/planned, 4 = intensive. However, Q43 deviates from this structure with only two possible values: 0 = no physician-prescribed procedures required, 4 = requires physician-prescribed procedures carried out by a licensed nurse. This anomaly eliminates in- termediate levels 1, 2, and 3, breaking the uniform scaling structure and potentially creating statistical modeling complications due to the bimodal distribution.

**Inconsistent Temporal Assessment Frameworks** The questionnaire employs multiple, in- compatible time frames across different assessment domains without clear justification for the temporal variations. Behavioral interventions are assessed over the “past 12 months,” emergency room visits use a “last year” timeframe, hospital admissions reference the “last six months,” med- ication changes examine the “past year,” while functional abilities assess “current status.” Some items fail to specify any temporal framework entirely. This temporal inconsistency complicates data interpretation and may introduce systematic bias when comparing support needs across different assessment domains.

**Special Scoring Rules Violating Ordinal Structure** Several questions employ automatic scoring rules that bypass the standard 0-4 ordinal scale, creating methodological inconsistencies. Q43 mandates an “automatic score of ’4’ if physician-prescribed procedures are required,” while Q36 includes a special provision that “anyone on Reglan/Metoclopramide, regardless of the reason, has this rating” of 4. These categorical override rules violate the ordinal measurement principles underlying the assessment instrument and may introduce artificial ceiling effects that distort the distribution of scores and compromise statistical modeling assumptions.

**Version Control and Documentation Issues** The questionnaire exhibits evidence of poor version control with conflicting information about revision dates, effective dates, and rule ref- erences. The document simultaneously references Version 4.0 as effective 2-15-08 and revised 5-21-15, while mentioning earlier versions with different scaling systems where “Level 5 that is now identical to Level 4.” Rule numbers and revision protocols appear inconsistent across different sections of the documentation. This suggests inadequate document management and quality assurance procedures that could lead to implementation inconsistencies across different assessment sites or time periods.

**Chapter 2**

**Previous Algorithm**

# Previous Algorithm

The current algorithm, designated as Model 5b, operates as a multiple linear regression model that calculates individual budget allocations based on a square-root transformation of fiscal year 2013-14 claims data. This approach incorporates 22 independent variables spanning living settings, age categories, and Questionnaire for Situational Information (QSI) assessment scores that evaluate behavioral, functional, and physical support needs. While the algorithm achieves an R-squared value of 0.7998, explaining approximately 80% of expenditure variation, this statistical performance comes with significant methodological concerns, including the removal of 9.40% of cases as outliers and reliance on data that is now over a decade old.

The temporal disconnect between the algorithm’s 2013-14 data foundation and current ser- vice delivery realities represents perhaps the most immediate concern. Over the intervening decade, disability services have experienced significant evolution in cost structures, service de- livery models, demographic patterns, and regulatory requirements. The algorithm’s inability to reflect these changes compromises its predictive validity and creates systematic biases that may disadvantage certain populations or service categories.

Beyond data currency issues, the algorithm’s statistical architecture raises fundamental ques- tions about its alignment with person-centered planning principles. The current approach pri- oritizes actuarial prediction based on historical patterns rather than incorporating individual preferences, goals, and strengths that form the cornerstone of person-centered planning. This disconnect between statistical methodology and philosophical foundation creates a system that may achieve statistical significance while failing to serve the individualization requirements that define quality disability services.

The outlier management approach presents additional concerns about the algorithm’s ability to serve the full spectrum of disability support needs. The requirement to remove nearly 10% of cases to achieve acceptable statistical fit suggests fundamental limitations in the model’s capacity to accommodate complex or atypical support requirements. This exclusion rate is particularly concerning given that individuals with the most intensive or unique needs may be precisely those most dependent on accurate algorithmic predictions for service access.

Variable validation limitations further compromise the algorithm’s comprehensiveness and potential compliance with statutory requirements. The exclusion of QSI questions 8 through 13 due to validation concerns creates gaps in needs assessment that may conflict with require- ments for thorough evaluation of individual characteristics and support needs. This limitation reflects broader challenges in balancing statistical rigor with comprehensive needs assessment in algorithmic systems.

# Statistical Methods Analysis

## Overview of Statistical Framework

The documentation of Model 5b presents the statistical methods employed in developing the Florida APD’s iBudget Algorithm. This section examines the technical approaches used for multiple linear regression modeling with transformations, model selection techniques, and outlier detection methods applied to predict APD consumers’ FY 2013-2014 expenditures.

## Multiple Linear Regression Foundation

The statistical framework builds upon linear regression as the primary analytical method for modeling relationships between dependent and independent variables. The study defines:

* **Dependent variable**: APD consumers’ FY 2013-2014 expenditures
* **Independent variables**: consumers’ age, living setting status, individual characteristics and support needs specified in QSI assessments

The classical multiple linear regression model is specified as:

*yi* = *β*0 + *β*1*x*1*i* + *β*2*x*2*i* + *· · ·* + *βpxpi* + *εi, i* = 1*,* 2*, . . . , n* (2.1) where *yi* represents the dependent variable, *{x*1*i, x*2*i, . . . , xpi}* are independent variables or predictors, *β*0 is the intercept, *β*0*, β*1*, . . . , βp* are unknown coefficients, and *ε*1*, ε*2*, . . . , εn* are random error terms.

*{ } { }*

## Statistical Assumptions and Limitations

The regression framework requires three critical assumptions for the random error terms:

1. Each term *εi* follows a normal distribution
2. Error terms *{ε*1*, ε*2*, . . . , εn}* are mutually independent
3. Each term *εi* has constant variance *σ*2 (homoscedasticity)

These assumptions present immediate challenges when applied to disability expenditure data, which typically exhibits high variability and non-normal distributions due to the diverse and individualized nature of support needs.

## Box-Cox Power Transformation

To address distributional concerns, the methodology employs Box-Cox power transformation to normalize the response variable. The transformation is defined as:

*z*(*λ*) =

*i*

*λ i*

*λ·GM* (*y*)*λ−*1

( *y −*1

if *λ ̸*= 0

(2.2)

where *GM* (*y*) = [IT*n*

*GM* (*y*) *·* ln(*yi*) if *λ* = 0

*y* ]1*/n* represents the geometric mean of observations. The scale ad-

*i*=1

*i*

justment by *GM* (*y*) ensures unit comparability across different transformation values.

The optimal transformation parameter *λ* is selected to minimize the Residual Sum of Squares:

*n*

*i*

*RSS*(*λ*) =

p*i*=1

*ε*ˆ(*λ*) 2

(2.3)

In practice, *RSS*(*λ*) is evaluated for discrete values:

*λ ∈ {−*3*, −*2*.*5*, −*2*, −*1*.*5*, −*1*, −*0*.*5*,* 0*,* 0*.*5*,* 1*,* 1*.*5*,* 2*,* 2*.*5*,* 3*}*.

## Model Selection via Bayesian Information Criterion

The methodology employs the Bayesian Information Criterion (SBC) for model selection among 125 candidate independent variables. This approach aims to identify variables with significant predictive power while maintaining model parsimony and avoiding overfitting.

The SBC framework compares multiple candidate models to select the configuration with op- timal predictive capability for individual budget allocation. However, this statistical optimization approach prioritizes mathematical fit over substantive considerations of individual needs assess- ment and person-centered planning principles.

## Methodological Concerns

The statistical framework reveals several fundamental limitations that impact the algorithm’s suitability for person-centered disability services:

### Outlier Management

The requirement to remove 9.40% of cases to achieve acceptable statistical fit indicates funda- mental model limitations in accommodating diverse support needs. This exclusion rate sug- gests the methodology cannot adequately serve individuals with complex or atypical require- ments—precisely those who may most depend on accurate algorithmic predictions.

### Temporal Validity

The reliance on FY 2013-2014 data for model development creates a significant temporal dis- connect with current service delivery realities. The statistical framework lacks mechanisms for updating or recalibrating the model to reflect evolving cost structures, service models, or demo- graphic patterns.

### Person-Centered Alignment

The emphasis on actuarial prediction based on historical patterns conflicts with person-centered planning principles that prioritize individual preferences, goals, and strengths. The statistical methodology treats individuals as data points to be fitted to historical patterns rather than unique persons with individualized support requirements.

## Implementation Implications

The statistical methods underlying Model 5b demonstrate technical competency within tra- ditional regression frameworks while revealing fundamental misalignment with contemporary disability services principles. The methodology’s focus on statistical optimization may achieve mathematical significance while failing to serve the individualization and person-centered re- quirements that define quality disability services.

The documented approach establishes that while the statistical framework follows accepted practices for regression modeling, its application to disability budget allocation raises signif- icant concerns about equity, individualization, and compliance with person-centered planning requirements. These methodological limitations provide important context for evaluating the algorithm’s overall suitability for Florida’s disability services system.

# Critical Deficiencies in Model 5b

Questions Q8, Q9, Q12, and Q13 were systematically excluded from statistical modeling because “items were not validated and the reliability of these items was not examined.” This represents a fundamental design flaw where questions addressing life changes and community inclusion were incorporated into the instrument without proper psychometric validation. The exclusion of these variables reduced the total usable predictors from 125 to a smaller subset, eliminating potentially valuable contextual information about life transitions and community participation that could influence support needs. This suggests inadequate instrument development protocols and quality control procedures.

The statistical analysis revealed multiple fundamental deficiencies that compromise the va- lidity and utility of the proposed linear regression models for resource allocation.

### Counter-Intuitive Negative Coefficients

The most egregious deficiency involved negative coefficients for the functional status sum (FSum) and physical status sum (PSum) variables in Model 5a1. These negative coefficients mathemat- ically implied that individuals with greater functional or physical impairments would receive *less* funding, directly contradicting the logical expectation that higher support needs should correspond to increased resource allocation. This fundamental violation of face validity forced the removal of these theoretically important variables from subsequent models, eliminating key predictors that should logically drive resource allocation decisions.

### Widespread Statistical Insignificance

Multiple predictor variables demonstrated non-significant relationships with the outcome vari- able, including disability type categories, individual QSI items, and interaction terms. For exam- ple, Q24 (evacuation ability) became non-significant (p-value = 0.53) after removing FSum and PSum from the model. Many disability type variables showed coefficients that were not statis- tically different from zero, despite their theoretical relevance to support needs. This pattern of insignificance suggests either inadequate model specification or fundamental measurement issues in the predictor variables.

### Violation of Distributional Assumptions

Residual diagnostic analysis revealed persistent deviations from normality assumptions even after square-root transformation of the dependent variable. The Q-Q normal plots demonstrated heavy tails inconsistent with the normal distribution required for valid linear regression inference. The diagnostic plots showed that ”the distribution is still away from the normal distribution in the two tails,” indicating that standard linear regression assumptions were not met, potentially invalidating hypothesis tests and confidence intervals.

### Excessive Outlier Exclusion

The final recommended Model 5b required exclusion of 9.40% of cases (2,410 consumers) as outliers—an extraordinarily high proportion suggesting either systematic data quality issues or fundamental model misspecification. Removing nearly one in ten cases raises serious concerns about model generalizability and may indicate that the linear modeling approach is fundamen- tally inappropriate for this data structure. Such extensive outlier removal suggests the presence of unmodeled nonlinear relationships or heteroscedasticity that the current approach cannot accommodate.

### Limited Construct Validity

Primary, secondary, and other disability type variables were ultimately excluded from the final model because they were ”not statistically predictive for the response variable.” The analysis noted that ”estimated coefficients for some categories of the three variables are negative and/or the estimated coefficients are not statistically different from zero.” This represents a critical fail- ure of construct validity, as disability type should logically influence support needs and resource requirements.

### Validation and Reliability Gaps

The exclusion of questions Q8, Q9, Q12, and Q13 due to lack of validation represents a significant methodological weakness. These items address life changes and community inclusion—factors that could substantially influence support needs. The systematic exclusion of unvalidated items,

while methodologically sound, highlights the incomplete development of the assessment instru- ment.

# Model 5b Implementation and Testing Framework

## Implementation Overview

The Model 5b algorithm has been implemented in Python as model5b.py, providing a complete computational framework for budget prediction based on the statistical methodology described in Section II. This implementation translates the regression coefficients from Table 4 of the UpdateStatisticalModelsiBudget document into a functional prediction system that can process individual QSI assessments and generate budget allocations according to the square-root trans- formation methodology.

The implementation maintains full fidelity to the original statistical model, including all 22 independent variables, interaction terms, and the critical square-root transformation that enables the algorithm to achieve its documented R-squared value of 0.7998. The program architecture emphasizes transparency, validation, and reproducibility, ensuring that predictions can be traced through each computational step.

## Program Execution

### System Requirements

The implementation requires Python 3.6 or higher with standard library modules only. No ex- ternal dependencies are required, ensuring compatibility across diverse computing environments. The program consists of two primary files:

* + - * + model5b.py - Complete Model 5b implementation
        + QSI-unit-test1.json - Comprehensive test dataset

### Execution Instructions

To execute the Model 5b test program, ensure both files are located in the same directory and run the following command:

python model5b.py

The program automatically loads the test dataset, processes all test cases through the Model 5b algorithm, and generates comprehensive output including individual predictions, summary statistics, and model performance metrics. No command-line arguments or configuration files are required for basic operation.

### Expected Output Structure

Program execution produces structured output organized into several sections:

1. **Loading Confirmation** - Verification of test data file access and case count
2. **Individual Predictions** - Detailed results for each test case including:
   * Individual identifier and demographic information
   * Living setting and age group classification
   * Predicted budget amount in dollars
   * Square-root scale intermediate calculation
3. **Summary Statistics** - Aggregate analysis including:
   * Count of successful predictions
   * Average, minimum, and maximum predicted budgets
   * Distribution characteristics across test cases
4. **Model Information** - Technical specifications including R-squared, outlier removal per- centage, and statistical performance metrics

## Test Dataset Specification

### Dataset Structure

The QSI-unit-test1.json file contains a comprehensive test dataset designed to validate Model 5b implementation across the full spectrum of disability support scenarios. The dataset employs JSON formatting for platform independence and includes both test cases and extensive metadata documentation.

The file structure consists of four primary components:

* + - * + **Metadata Section** - Dataset description, version information, and data source documen- tation
        + **Test Cases Array** - Twelve individual assessment records representing diverse support scenarios
        + **Variable Definitions** - Complete specification of all input variables and their valid ranges
        + **Model Information** - Technical parameters and performance characteristics of Model 5b
      1. **Test Case Coverage**

The dataset includes twelve carefully constructed test cases that systematically cover the param- eter space defined by Model 5b variables:

**Living Setting Distribution:**

* + - * + Family Home (FH): 4 cases representing the reference level
        + Independent Living & Supported Living (ILSL): 3 cases with varying support intensities
        + Residential Habilitation Standard (RH1): 2 cases including standard residential care
        + Residential Habilitation Behavior Focus (RH2): 1 case with behavioral specialization
        + Residential Habilitation Intensive Behavior (RH3): 1 case with intensive behavioral support
        + Residential Habilitation Special Medical (RH4): 1 case with complex medical needs

**Age Group Representation:**

* + - * + Under 21 (reference level): 2 cases representing adolescent populations
        + Age 21-30: 4 cases covering young adult transition period
        + Age 31+: 6 cases spanning adult and senior populations (ages 31-67)

**Support Need Variation:** The test cases systematically vary across support intensity levels:

* + - * + **Minimal Support** (TEST007): Teenager with limited intervention requirements
        + **Moderate Support** (TEST001, TEST008, TEST011): Individuals with balanced func- tional and behavioral needs
        + **High Support** (TEST003, TEST010): Adults requiring intensive assistance across mul- tiple domains
        + **Severe Support** (TEST005, TEST006): Complex cases with maximum intervention re- quirements
      1. **Variable Validation Framework**

Each test case includes validation of all required Model 5b input variables:

**Demographic Variables:**

* + - * + living setting - Categorical variable with six valid levels
        + age - Continuous variable determining age group classification

**QSI Sum Scores:**

* + - * + bsum - Behavioral status sum (0-24 range)
        + fsum - Functional status sum (0-44 range)
        + psum - Physical status sum (0-76 range)

**Individual QSI Questions:** Ten specific questions (Q16, Q18, Q20, Q21, Q23, Q28, Q33, Q34, Q36, Q43) each scored on 0-4 scales representing:

* + - * + Functional domains: Eating, transfers, hygiene, dressing, self-protection
        + Behavioral domains: Sexual behavior, aggression, restraint use
        + Medical domains: Psychotropic medications, physician-prescribed treatments
      1. **Data Integrity and Realism**

The test dataset maintains realistic relationships between variables, ensuring that sum scores align with individual question responses and that support needs correspond appropriately to living settings. For example, individuals in RH3 and RH4 settings demonstrate correspondingly higher QSI scores, while those in family homes show more variable support profiles reflecting diverse family capacity.

The dataset also incorporates edge cases and boundary conditions to test algorithm robust- ness, including individuals with minimal support needs, maximum scoring scenarios, and atypical combinations that may occur in real-world assessments.

* + 1. **Implementation Validation**
       1. **Computational Accuracy**

The Python implementation reproduces the exact coefficient structure documented in the Updat- eStatisticalModelsiBudget report, ensuring mathematical fidelity to the research methodology. All regression coefficients, interaction terms, and transformation procedures match the specifi- cations in Table 4, enabling direct comparison with the original statistical analysis.

The program includes comprehensive input validation to prevent computational errors and ensure that all QSI scores fall within their defined ranges. Invalid inputs generate descriptive error messages identifying the specific validation failure, supporting quality assurance in operational deployment.

* + - 1. **Transparency and Traceability**

Each prediction includes detailed intermediate calculations, allowing users to trace the contri- bution of individual variables to the final budget prediction. The output format displays the square-root scale calculation before transformation to dollars, enabling verification of the math- ematical procedures against manual calculations.

The implementation also provides complete documentation of which coefficients were applied for each individual, including living setting classification, age group determination, and specific QSI question contributions. This transparency supports both technical validation and policy analysis of algorithmic decision-making.

* + 1. **Testing Framework Applications**

This implementation and testing framework serves multiple analytical purposes beyond basic algorithm validation:

**Policy Analysis:** The comprehensive test cases enable examination of how different policy scenarios (changes in living setting availability, age group definitions, or QSI scoring protocols) would affect budget predictions across diverse populations.

**Equity Assessment:** The systematic coverage of demographic and support need combina- tions facilitates analysis of potential disparities in budget allocation across different population subgroups.

**Sensitivity Analysis:** The modular implementation structure supports investigation of how changes to individual coefficients or variable definitions would propagate through the prediction system.

**Validation Studies:** The test framework provides a standardized basis for comparing alter- native algorithmic approaches or validating implementation accuracy across different program- ming environments.

The combination of comprehensive implementation and systematic test data establishes a robust foundation for ongoing analysis and refinement of the Florida APD iBudget algorithm methodology.

|  |  |  |
| --- | --- | --- |
|  | **2.4.6** | **Source code & Output** |
| **Python** | **Implementation:** |
| 1 | #!/ usr/ | bin / env python 3 |
| 2 | """ |  |
| 3 | Model 5 | b Implementation for Florida APD iBudget Algorithm |
| 4 |  |  |

5 This module implements the final Model 5 b from the Update StatisticalModelsiBudget document.

6 The model uses square - root transformation and multiple linear regression to predict

7 individual budget allocations based on QSI assessment data .

8

9 Model 5 b uses the following coefficients ( from Table 4):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | - | Intercept: 27 .5720 | | | |
| 11 | - | Living Settings: ILSL (35 .8220 ) , RH1 (90 .6294 ) , RH2 (131 .7576 ) , RH3 | | | |
|  |  | (209 .4558 ) , RH4 (267 .0995 ) | | | |
| 12 | - | Age Groups: Age21 -30 (47 .8473 ) , Age31 + (48 .9634 ) | | | |
| 13 | - Behavioral/ Functional Sums: BSum | | (0 .4954 ) , | FHFS | um (0 .6349 ) , SLFSum |
|  | (2 .0529 ) , SLBSum (1 .4501 ) | |  |  |  |
| 14 | - QSI Questions: Q16 (2 .4984 ) , Q18 | | (5 .8537 ) , | Q20 | (2 .6772 ) , Q21 (2 .7878 ) , |
|  | Q23 (6 .3555 ) , | |  |  |  |
| 15 | Q28 (2 .2803 ) , Q33 | | (1 .2233 ) , | Q34 | (2 .1764 ) , Q36 (2 .6734 ) , |
|  | Q43 (1 .9304 ) | |  |  |  |

16

17 Reference levels ( coefficients = 0):

18 - Living Setting : Family Home ( FH)

19 - Age: Under 21

20 """

21

22 import json

23 import math

24 import sys

25 from typing import Dict , Any , Optional

26 from dataclasses import dataclass

27 from datetime import datetime

28

29

30 @ dataclass

31 class Model5 b Coefficients :

32 """ Model 5 b regression coefficients from the final algorithm ."""

33

34 # Intercept

35 intercept: float = 27 .5720

36

37 # Living Setting coefficients ( FH is reference level with 0)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 38 | live\_ilsl: float |  | = 35 .8220 | # | Independent | Living & Suppo | rted Living |
| 39 | live\_rh 1 : float | = | 90 .6294 | # | Residential | Habilitation , | Standard and |
|  | Live In |  |  |  |  |  |  |
| 40 | live\_rh 2 : float | = | 131 .7576 | # | Residential | Habilitation , | Behavior |
|  | Focus |  |  |  |  |  |  |
| 41 | live\_rh 3 : float | = | 209 .4558 | # | Residential | Habilitation , | Intensive |
|  | Behavior |  |  |  |  |  |  |
| 42 | live\_rh 4 : float | = | 267 .0995 | # | Residential | Habilitation , | CTEP and |

Special Medical Home Care

43

44 # Age Group coefficients ( Under 21 is reference level with 0)

45 age\_21 \_30 : float = 47 .8473 # Age 21 -30

46 age\_31 \_plus: float = 48 .9634 # Age 31+

47

48 # Sum and interaction coefficients

49 bsum : float = 0 .4954 # Behavioral status sum score

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 50  51 | fhfsum : float = 0 .6349  interaction  slfsum : float = 2 .0529 | | | | #  # | Family Home by Functional  ILSL by Functional status | | status  interaction |
| 52 | slbsum : float = 1 .4501 | | | | # | ILSL by Behavioral status | | interaction |
| 53 |  | | | |  |  | |  |
| 54 | # QSI Question coefficients | | | |  |  | |  |
| 55 | q16 : float = 2 .4984 | | | | # | Eating | |  |
| 56 | q18 : float = 5 .8537 | | | | # | Transfers | |  |
| 57 | q20 : float = 2 .6772 | | | | # | Hygiene | |  |
| 58 | q21 : float = 2 .7878 | | | | # | Dressing | |  |
| 59 | q23 : float = 6 .3555 | | | | # | Self - protection | |  |
| 60 | q28 : float = 2 .2803 | | | | # | Inappropriate Sexual Behav | | ior |
| 61 | q33 : | float | = | 1 .2233 | # | Injury | to Person Caused by Aggression | |
| 62 | q34 : | float | = | 2 .1764 | # | Use of | Mechanical Restraints | |
| 63 | q36 : | float | = | 2 .6734 | # | Use of | Psychotropic Medications | |
| 64 | q43 : | float | = | 1 .9304 | # | Treatme | nt ( Physician Prescribed ) | |
| 65 |  |  |  |  |  |  |  | |
| 66 |  |  |  |  |  |  |  | |

67 class Tee Output:

68 """

69 Helper class to write output to both console and file simultaneously .

70 """

71 def init ( self , filename ):

72 self. terminal = sys. stdout

73 self. log = open ( filename , ’w’)

74

75 def write ( self , message ):

76 self. terminal. write ( message )

77 self. log. write ( message )

78

79 def flush ( self):

80 self. terminal. flush ()

81 self. log. flush ()

82

83 def close ( self):

84 self. log. close ()

85

86

87 class Model5 b :

88 """

89 Implementation of Model 5 b for Florida APD iBudget Algorithm .

90

91 This class implements the final regression model with square - root transformation

92 that achieved R- squared = 0 .7998 after removing 9.40% outliers.

93 """

94

95 def init ( self):

96 self. coefficients = Model5 b Coefficients ()

97 self. model\_info = {

98 " name ": " Model 5 b",

99 " r\_squared ": 0.7998 ,

100 " outliers\_removed ": 0.094 ,

101 " residual\_standard\_error ": 30.82 ,

102 " degrees\_of\_freedom ": 23193 ,

|  |  |  |  |
| --- | --- | --- | --- |
| 103 |  | " f\_statistic": 4412 , | |
| 104 |  | " p\_value ": " < 2.2 e -16 " | |
| 105 |  | } | |
| 106 |  |  | |
| 107 | def | validate\_input ( self , qsi\_data : Dict[ str , Any ]) -> Dict[ str , Any ]: | |
| 108 |  | """ | |
| 109 |  | Validate and normalize QSI input data . | |
| 110 |  |  | |
| 111 |  | Args: | |
| 112 | qsi\_data : Dictionary containing QSI assessment data | | |
| 113 |  | | |
| 114 | Returns: | | |
| 115 | Validated and normalized data dictionary | | |
| 116 |  | | |
| 117 | Raises: | | |
| 118 | Value Error: | | If required fields are missing or invalid |
| 119 | """ | |  |
| 120  121 | required\_fields  ’]  qsi\_questions = | | = [’ living\_setting ’, ’ age ’, ’ bsum ’, ’ fsum ’, ’ psum  [’ Q16 ’, ’ Q18 ’, ’ Q20 ’, ’ Q21 ’, ’ Q23 ’, ’ Q28 ’, ’ Q33 ’, |
|  | ’ Q34 ’, ’ Q36 ’, ’ Q43 ’] | | |
| 122 |  | | |
| 123 | # Check required fields | | |
| 124 | for field in required\_fields : | | |
| 125 | if field not in qsi\_data : | | |

126 raise Value Error( f" Missing required field : { field }")

127

128 # Check QSI questions

129 for q in qsi\_questions :

130 if q not in qsi\_data :

131 raise Value Error( f" Missing required QSI question : { q}")

132

133 # Validate living setting

134 valid\_living\_settings = [’ FH ’, ’ ILSL ’, ’ RH1 ’, ’ RH2 ’, ’ RH3 ’, ’ RH4 ’

]

135 if qsi\_data [’ living\_setting ’] not in valid\_living\_settings :

136 raise Value Error( f" Invalid living\_setting . Must be one of: { valid\_living\_settings }")

137

138 # Validate age

139 if not isinstance ( qsi\_data [’ age ’], ( int , float)) or qsi\_data [’ age ’] < 0:

140 raise Value Error(" Age must be a non - negative number")

141

142 # Validate QSI scores (0 -4 scale )

143 for q in qsi\_questions :

144 score = qsi\_data [ q]

145 if not isinstance ( score , ( int , float)) or score < 0 or score

> 4:

146 raise Value Error( f"{ q} must be between 0 and 4 , got: { score }")

147

148 # Validate sum scores

149 if not (0 <= qsi\_data [’ bsum ’] <= 24): # 6 questions by 4 max score

150 raise Value Error(" BSum must be between 0 and 24 ")

151 if not (0 <= qsi\_data [’ fsum ’] <= 44): # 11 questions by 4 max score

152 raise Value Error(" FSum must be between 0 and 44 ")

153 if not (0 <= qsi\_data [’ psum ’] <= 76): # 19 questions by 4 max score

154 raise Value Error(" PSum must be between 0 and 76 ")

155

tion\_terms ( self , qsi\_data : Dict[ str , Any ]) ->

|  |  |
| --- | --- |
| 156 | return qsi\_data |
| 157 |  |
| 158 | def calculate\_interac |
| 159 | Dict[ str , float ]:  """ |

160 Calculate interaction terms between living setting and sum scores

.

161

162 Args:

163 qsi\_data : Validated QSI data

164

165 Returns:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 166 | Dictionary containing interaction term values | | | | |
| 167 | """ | | | | |
| 168 | living\_setting = qsi\_data [’ living\_setting ’] | | | | |
| 169 | fsum = qsi\_data [’ fsum ’] | | | | |
| 170 | bsum = qsi\_data [’ bsum ’] | | | | |
| 171 |  | | | | |
| 172 | interactions = { | | | | |
| 173 |  | ’ fhfsum ’: | 0 , | # | Family Home by Functional Sum |
| 174 |  | ’ slfsum ’: | 0 , | # | ILSL by Functional Sum |
| 175 |  | ’ slbsum ’: | 0 | # | ILSL by Behavioral Sum |
| 176 | } |  |  |  |  |
| 177 |  |  |  |  |  |

178 if living\_setting == ’ FH ’:

179 interactions[’ fhfsum ’] = fsum

180 elif living\_setting == ’ ILSL ’:

181 interactions[’ slfsum ’] = fsum

182 interactions[’ slbsum ’] = bsum

183

184 return interactions

185

186 def predict\_square\_root\_scale ( self , qsi\_data : Dict[ str , Any ]) -> float:

187 """

188 Calculate prediction in square - root scale using Model 5 b coefficients .

189

190 Args:

ata

|  |  |
| --- | --- |
| 191 | qsi\_data : Validated QSI assessment d |
| 192 |  |
| 193 | Returns: |
| 194 | Predicted value in square - root scale |
| 195 | """ |
| 196 | # Start with intercept |
| 197 | prediction = self. coefficients. intercept |
| 198 |  |

199 # Add living setting effects ( FH is reference level)

200 living\_setting = qsi\_data [’ living\_setting ’]

201 if living\_setting == ’ ILSL ’:

202 prediction += self. coefficients. live\_ilsl

203 elif living\_setting == ’ RH1 ’:

204 prediction += self. coefficients. live\_rh 1

205 elif living\_setting == ’ RH2 ’:

206 prediction += self. coefficients. live\_rh 2

207 elif living\_setting == ’ RH3 ’:

208 prediction += self. coefficients. live\_rh 3

209 elif living\_setting == ’ RH4 ’:

210 prediction += self. coefficients. live\_rh 4

211 # FH has coefficient 0 ( reference level)

|  |  |  |
| --- | --- | --- |
| 212 |  | |
| 213 | # Add | age effects ( Under 21 is reference level) |
| 214 | age = | qsi\_data [’ age ’] |
| 215 | if 21 | <= age <= 30: |

216 prediction += self. coefficients. age\_21 \_30

217 elif age >= 31:

218 prediction += self. coefficients. age\_31 \_plus

219 # Under 21 has coefficient 0 ( reference level)

220

221 # Add behavioral sum effect

222 prediction += self. coefficients. bsum \* qsi\_data [’ bsum ’]

223

224 # Add interaction terms

225 interactions = self. calculate\_interaction\_terms ( qsi\_data )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 226 | prediction | | += | self. coefficients. fhfs | um \* interactions[’ fhfsum ’] |
| 227 | prediction | | += | self. coefficients. slfs | um \* interactions[’ slfsum ’] |
| 228 | prediction | | += | self. coefficients. slbs | um \* interactions[’ slbsum ’] |
| 229 |  | |  |  |  |
| 230 | # Add QSI | | ques | tion effects |  |
| 231 | prediction | | += | self. coefficients. q16 | \* qsi\_data [’ Q16 ’] |
| 232 | prediction | | += | self. coefficients. q18 | \* qsi\_data [’ Q18 ’] |
| 233 | prediction | | += | self. coefficients. q20 | \* qsi\_data [’ Q20 ’] |
| 234 | prediction | | += | self. coefficients. q21 | \* qsi\_data [’ Q21 ’] |
| 235 | prediction | | += | self. coefficients. q23 | \* qsi\_data [’ Q23 ’] |
| 236 | prediction | | += | self. coefficients. q28 | \* qsi\_data [’ Q28 ’] |
| 237 | prediction | | += | self. coefficients. q33 | \* qsi\_data [’ Q33 ’] |
| 238 | prediction | | += | self. coefficients. q34 | \* qsi\_data [’ Q34 ’] |
| 239 | prediction | | += | self. coefficients. q36 | \* qsi\_data [’ Q36 ’] |
| 240 | prediction | | += | self. coefficients. q43 | \* qsi\_data [’ Q43 ’] |
| 241 |  | |  |  |  |
| 242 |  | return prediction | | | |
| 243 |  |  | | | |
| 244 | def | predict\_budget ( self , qsi\_data : Dict[ str , Any ]) -> Dict[ str , Any ]: | | | |
| 245 |  | """ | | | |
| 246 |  | Predict individual budget allocation using Model 5 b. | | | |
| 247 |  |  | | | |
| 248 |  | Args: | | | |
| 249 |  | qsi\_data : QSI assessment data | | | |
| 250 |  |  | | | |
| 251 |  | Returns: | | | |
| 252 |  | Dictionary containing prediction results | | | |
| 253 |  | """ | | | |

|  |  |  |
| --- | --- | --- |
| 254  255  256  257  258  259  260  261  262  263  264  265 | # Validate input  validated\_data = self. validate\_input ( qsi\_data )  # Calculate prediction in square - root scale  sqrt\_prediction = self. predict\_square\_root\_scale ( validated\_data )  # Transform back to dollar scale by squaring budget\_prediction = sqrt\_prediction \*\* 2  # Calculate interaction terms for transparency  interactions = self. calculate\_interaction\_terms ( validated\_data ) | |
| 266 |  | return { |
| 267 |  | ’ predicted\_budget ’: round ( budget\_prediction , 2), |
| 268 |  | ’ sqrt\_scale\_prediction ’: round ( sqrt\_prediction , 4), |
| 269 |  | ’ model\_info ’: self. model\_info , |
| 270 |  | ’ input\_data ’: validated\_data , |
| 271 |  | ’ interaction\_terms ’: interactions , |
| 272 |  | ’ coefficients\_used ’: { |
| 273 |  | ’ living\_setting ’: validated\_data [’ living\_setting ’], |
| 274 |  | ’ age\_group ’: self. \_get\_age\_group ( validated\_data [’ age ’]), |
| 275  276 |  | ’ qsi\_scores ’: { q: validated\_data [ q] for q in [’ Q16 ’, ’ Q18  ’, ’ Q20 ’, ’ Q21 ’, ’ Q23 ’, ’ Q28 ’, ’ Q33 ’, ’ Q34 ’, ’ Q36 ’, ’ Q43 ’]}  } |
| 277 |  | } |
| 278 |  |  |
| 279 | def | \_get\_age\_group ( self , age: float) -> str: |
| 280 |  | """ Helper function to determine age group .""" |
| 281 |  | if age < 21: |
| 282 |  | return " Under 21 ( reference )" |
| 283 |  | elif 21 <= age <= 30: |
| 284 |  | return " 21 -30 " |
| 285 |  | else : |
| 286 |  | return " 31+ " |
| 287 |  |  |
| 288 | def | predict\_batch ( self , qsi\_data\_list : list) -> list: |
| 289 |  | """ |
| 290 |  | Predict budgets for multiple individuals. |
| 291 |  |  |
| 292 |  | Args: |
| 293 | qsi\_data\_list : List of QSI assessment data dictionaries | |
| 294 |  | |
| 295 | Returns: | |

296

297

"""

List of prediction results

298 results = []

299 for i, qsi\_data in enumerate ( qsi\_data\_list ):

300 try :

301 result = self. predict\_budget ( qsi\_data )

302 result[’ record\_index ’] = i

303 results. append ( result)

304 except Exception as e:

305 results. append ({

306 ’ record\_index ’: i,

307 ’ error ’: str( e),

308 ’ input\_data ’: qsi\_data

309 })

310 return results

311

312

313 def main ():

314 """

315 Main function to test Model 5 b implementation using QSI - unit - test1 . json

316 Output is written to both console and model5 b\_output . txt

317 """

318 # Set up dual output to console and file

319 output\_filename = ’ model5 b\_output . txt’

320 tee = Tee Output( output\_filename )

321 original\_stdout = sys. stdout

322 sys. stdout = tee

323

324 try :

325 # Add timestamp to output

326 print(" Florida APD iBudget Algorithm - Model 5 b Implementation ")

327 print(" =" \* 60)

328 print( f" Execution Date / Time : { datetime . now (). strftime (’% Y -% m -% d % H:% M:% S ’)}")

329 print( f" Output File : { output\_filename }")

330 print(" =" \* 60)

331

332 # Initialize the model

333 model = Model5 b ()

334

335 try :

336 # Load test data

337 with open (’QSI - unit - test1 . json ’, ’r’) as f:

338 test\_data = json . load ( f)

339

340 print( f"\ n Loaded { len ( test\_data [’ test\_cases ’])} test cases from QSI - unit - test1 . json ")

341 print( f" Test data description : { test\_data [’ description ’]}")

342

343 # Run predictions

344 results = model. predict\_batch ( test\_data [’ test\_cases ’])

345

346 # Display results

347 print( f"\ n Model 5 b Prediction Results:")

348 print("-" \* 40)

349

350 for result in results:

351 if ’ error ’ in result:

352 print( f" Record { result[’ record\_index ’]}: ERROR - { result[’ error ’]}")

353 else :

354 data = result[’ input\_data ’]

355 print( f"\ n Record { result[’ record\_index ’]}:")

356 print( f" Individual: { data . get(’ individual\_id ’, ’N/ A ’)}")

357

print( f" Living Setting : { data [’ living\_setting ’]}")

print( f" Age: { data [’ age ’]} ({ result[’ coefficients\_used ’][’ age\_group ’]})")

print( f" Predicted Budget: ${ result[’ predicted\_budget ’]:,.2 f}")

print( f" Square - root Scale : { result[’ sqrt\_scale\_prediction ’]}")

# Summary statistics

successful\_predictions = [ r for r in results if ’ error ’ not in r]

if successful\_predictions :

budgets = [ r[’ predicted\_budget ’] for r in successful\_predictions ]

print( f"\ n Summary Statistics:") print( f" Successful predictions: { len (

successful\_predictions )}")

print( f" Average predicted budget: ${ sum ( budgets)/ len ( budgets):,.2 f}")

print( f"

)

print( f"

)

Minimum predicted

budget: ${ min ( budgets):,.2 f}"

Maximum predicted

budget: ${ max( budgets):,.2 f}"

print( f"\ n Model Information :")

print( f" R- squared : { model. model\_info [’ r\_squared ’]}") print( f" Outliers removed : { model. model\_info [’

outliers\_removed ’]\*100:.1 f}%")

print( f" Residual standard error: { model. model\_info [’ residual\_standard\_error ’]}")

print( f"\ n" + " =" \* 60)

print( f" Execution completed successfully .") print( f" Results saved to: { output\_filename }")

except File NotFound Error :

print("\ n Error: QSI - unit - test1 . json not found .") print(" Please ensure the test data file is in the same

directory .")

except json . JSONDecode Error as e: print( f"\ n Error reading JSON file : { e}")

except Exception as e: print( f"\ n Unexpected error: { e}")

finally :

# Restore original stdout and close file sys. stdout = original\_stdout

tee. close ()

print( f"\ n Output has been written to both console and { output\_filename }")

if name == " main ":

main ()

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

**Output:**

Florida APD iBudget Algorithm - Model 5 b Implementation

============================================================

Execution Date / Time : 2025 -09 -05 17 :40 :02 Output File : model5 b\_output . txt

============================================================

Loaded 12 test cases from QSI - unit - test1 . json

Test data description : Unit test data for

based on QSI assessments

Model 5 b iBudget

Algorithm

Model 5 b Prediction Results:

----------------------------------------

Record 0:

Individual: TEST 001 Living Setting : ILSL Age: 25 (21 -30)

Predicted Budget: $42 ,960 .19 Square - root Scale : 207 .2684

Record 1:

Individual: TEST 002 Living Setting : FH

Age: 19 ( Under 21 ( reference )) Predicted Budget: $4 ,709 .91 Square - root Scale : 68 .6288

Record 2:

Individual: TEST 003 Living Setting : RH1 Age: 35 (31+)

Predicted Budget: $69 ,109 .36 Square - root Scale : 262 .8866

Record 3:

Individual: TEST 004 Living Setting : RH2 Age: 28 (21 -30)

Predicted Budget: $96 ,521 .94 Square - root Scale : 310 .6798

Record 4:

Individual: TEST 005 Living Setting : RH3 Age: 42 (31+)

Predicted Budget: $169 ,866 .55 Square - root Scale : 412 .1487

Record 5:

Individual: TEST 006 Living Setting : RH4 Age: 55 (31+)

Predicted Budget: $215 ,268 .90 Square - root Scale : 463 .9708

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55 Record 6:

56 Individual: TEST 007

57 Living Setting : FH

58 Age: 16 ( Under 21 ( reference ))

59 Predicted Budget: $3 ,662 .69

60 Square - root Scale : 60 .5202

61

62 Record 7:

63 Individual: TEST 008

64 Living Setting : ILSL

65 Age: 31 (31+)

66 Predicted Budget: $56 ,536 .28

67 Square - root Scale : 237 .7736

68

69 Record 8:

70 Individual: TEST 009

71 Living Setting : FH

72 Age: 24 (21 -30)

73 Predicted Budget: $19 ,418 .14

74 Square - root Scale : 139 .349

75

76 Record 9:

77 Individual: TEST 010

78 Living Setting : RH1

79 Age: 67 (31+)

80 Predicted Budget: $68 ,804 .18

81 Square - root Scale : 262 .3055

82

83 Record 10:

84 Individual: TEST 011

85 Living Setting : ILSL

86 Age: 29 (21 -30)

87 Predicted Budget: $40 ,415 .35

88 Square - root Scale : 201 .0357

89

90 Record 11:

91 Individual: TEST 012

92 Living Setting : FH

93 Age: 38 (31+)

94 Predicted Budget: $27 ,697 .45

95 Square - root Scale : 166 .4255

96

97 Summary Statistics:

98 Successful predictions: 12

99 Average predicted budget: $67 ,914 .24

100 Minimum predicted budget: $3 ,662 .69

101 Maximum predicted budget: $215 ,268 .90

102

103 Model Information :

104 R- squared : 0 .7998

105 Outliers removed : 9.4%

106 Residual standard error: 30.82

107

108 ============================================================

109

Execution completed successfully .

Results saved to: model5 b\_output . txt

110

**Chapter 3**

**Alternative Algorithms**

# Alternative Algorithms

The proposed alternative algorithms represent six distinct categories of quantitative approaches, each designed to address specific limitations in the current system while advancing compliance with person-centered planning requirements. Enhanced linear regression approaches focus on updating data sources, improving outlier management, and expanding variable inclusion while maintaining the interpretability advantages of traditional statistical methods. Machine learn- ing ensemble approaches leverage advanced algorithms to capture non-linear relationships and complex interactions while providing transparency through feature importance analysis and pre- diction explanation techniques.

Hybrid statistical-clinical approaches represent a fundamental reconceptualization of algorith- mic design, combining statistical prediction with explicit mechanisms for incorporating clinical judgment and person-centered planning elements. These approaches acknowledge that purely statistical methods may be insufficient for capturing the full complexity of individual needs and preferences that effective disability services require.

Person-centered optimization approaches directly address compliance requirements by for- mulating budget allocation as a multi-objective optimization problem that balances statistical accuracy with goal alignment and fairness considerations. These methods represent a paradigm shift from prediction-focused algorithms to optimization-focused systems that explicitly incor- porate individual preferences and societal equity concerns into the mathematical formulation. Modern time-aware approaches address temporal validity concerns through dynamic regres- sion methods that adapt coefficients over time and longitudinal models that track individual trajectories. These approaches recognize that both population-level service patterns and indi- vidual needs evolve over time, requiring algorithmic systems that can adapt rather than remain static.

Specialized needs-based approaches acknowledge the heterogeneity within disability popula- tions through mixture models that identify distinct subpopulations and support vector regression methods that can accommodate high-dimensional assessment data and non-linear relationships. These approaches recognize that one-size-fits-all algorithms may be inherently inadequate for serving diverse disability populations with varying support requirements and preferences.

## Advanced Mathematical and Statistical Modeling Approaches

Given the identified deficiencies, several advanced modeling approaches could substantially im- prove the analysis of QSI data while addressing the fundamental limitations of the current linear regression framework.

### Regularization Methods for High-Dimensional Data

The QSI dataset presents a high-dimensional modeling challenge with 125 potential predictors and complex multicollinearity among related assessment items. Regularization methods provide principled approaches to variable selection and coefficient estimation.

**LASSO Regression (L1 Regularization)** LASSO regression addresses the variable selection problem through automatic feature selection:

*β*ˆ*LASSO* = arg min {*∥y − Xβ∥*2 + *λ∥β∥*1} (3.1)

*β*

2

where *λ* controls the sparsity penalty, automatically setting irrelevant coefficients to exactly

zero. This approach would eliminate the need for ad-hoc variable removal while providing a principled method for identifying the most predictive QSI items.

**Ridge Regression (L2 Regularization)** Ridge regression addresses multicollinearity among QSI items without variable elimination:

*β*ˆ*Ridge* = arg min {*∥y − Xβ∥*2 + *λ∥β∥*2} (3.2)

*β*

2

2

This approach shrinks correlated coefficients toward each other, potentially resolving the

negative coefficient problem by stabilizing parameter estimates.

**Elastic Net Regularization** Elastic Net combines both L1 and L2 penalties to simultaneously address variable selection and multicollinearity:

*β*ˆ*EN* = arg min {*∥y − Xβ∥*2 + *λ*1*∥β∥*1 + *λ*2*∥β∥*2} (3.3)

*β*

2

2

Given the natural groupings in QSI data, this approach could identify relevant subsets of

questions while maintaining stable coefficient estimates.

### Sparse Estimation Techniques

**Adaptive LASSO** Adaptive LASSO incorporates data-driven weights to reduce bias in coef- ficient estimation:

*β*ˆ*AL* = arg min

*β*



*∥y − Xβ∥*2 + *λ*

p*j*=1

*p*

*wj|βj|*



(3.4)

where weights *wj* = 1*/ β*ˆ(0) *γ* are based on initial consistent estimates, potentially addressing the coefficient sign problems observed in the current models.

2

*j*

*| |*

**Group LASSO** Given the natural grouping of QSI items into functional, behavioral, and physical domains, Group LASSO enables selection of entire groups:

2

*G*

*β*ˆ*GL* = arg min

*β*

(*∥y − Xβ∥*2 + *λ*

*g*p=1

*√pg∥βg∥*2)

(3.5)

This approach could determine whether entire assessment domains should be included or excluded from the allocation algorithm.

### Robust Regression Approaches

To address the outlier problem without arbitrary data exclusion, robust regression methods provide principled alternatives.

**M-Estimation** M-estimators minimize robust loss functions:

*n*

*− x β*

*T*

*β*ˆ*M* = arg min

*β*

p*i*=1

*ρ yi* *i*

*σ*

(3.6)

Using Huber or Tukey bisquare loss functions *ρ*( ) that downweight extreme observations rather than excluding them entirely.

*·*

**Quantile Regression** Given the apparent heteroscedasticity and non-normal residuals, quan- tile regression models conditional quantiles rather than means:

*β*ˆ*τ* = arg min p *ρτ* (*yi − xT β*) (3.7)

*n*

*−*

*β*

*i*=1

*i*

where *ρτ* (*u*) = *u*(*τ* **1***u<*0) is the quantile loss function. This approach could model different resource allocation patterns across the support needs distribution.

### Machine Learning Approaches for Nonlinear Relationships

**Random Forest Regression** Random Forest can capture complex nonlinear relationships and interactions among QSI items:

*f*ˆ(*x*) = 1 p *T* (*x*) (3.8)

*B*

*B*

*b*=1

*b*

where *Tb* represents individual decision trees trained on bootstrap samples. This approach

provides variable importance measures and handles interactions naturally.

**Gradient Boosting** Gradient boosting sequentially builds weak learners to minimize predic- tion error:

*Fm*(*x*) = *Fm−*1(*x*) + *γmhm*(*x*) (3.9) where *hm* minimizes the residual from iteration *m* 1. This approach could identify complex patterns in the QSI data that linear models cannot capture.

*−*

### Ordinal Regression Methods

Given the ordinal nature of QSI responses (0-4 scale), proportional odds models may be more appropriate than treating the data as continuous:

logit(*P* (*Y ≤ j|x*)) = *αj − xT β* (3.10) This approach respects the ordinal structure of the assessment data while potentially improv- ing model fit.

### Hierarchical and Mixed-Effects Models

To account for potential clustering within service areas or provider organizations:

*yij* = *Xijβ* + *Zijbi* + *ϵij* (3.11)

where *bi N* (0*, D*) represents random effects for cluster *i*. This approach could account for systematic differences in resource allocation patterns across regions or providers.

*∼*

### Ensemble Methods

**Model Stacking** Stacking combines multiple base models using a meta-learner:

p

*K*

*y*ˆ = *α*0 + *αkf*ˆ*k* (*x*) (3.12)

*k*=1

where *f*ˆ*k* (*x*) represents predictions from different modeling approaches, potentially combining the strengths of parametric and nonparametric methods.

**Bayesian Model Averaging** BMA incorporates model uncertainty into predictions:

p

*K*

*y*ˆ = *P* (*Mk|*data) *· f*ˆ*k* (*x*) (3.13)

*k*=1

providing principled uncertainty quantification for resource allocation decisions.

## Recommendations for Model Development

Given the complexity of the QSI data and the fundamental deficiencies in the current approach, we recommend a multi-stage modeling strategy:

1. **Baseline Establishment**: Implement cross-validated elastic net regression as a regular- ized linear baseline, addressing multicollinearity and variable selection issues.
2. **Nonlinear Enhancement**: Apply gradient boosting to detect and model nonlinear rela- tionships and interactions among QSI variables.
3. **Robustness Testing**: Evaluate quantile regression and robust methods to assess sensi- tivity to distributional assumptions and outliers.
4. **Ensemble Integration**: Combine multiple approaches using stacking or Bayesian model averaging to leverage the strengths of different methodological frameworks.
5. **Validation Framework**: Implement rigorous cross-validation and holdout testing to en- sure model generalizability and prevent overfitting.

This comprehensive approach would address the identified deficiencies while fully utilizing the rich multidimensional structure of the QSI assessment data, providing a more reliable foundation for equitable resource allocation decisions.

# Current Algorithm Analysis

## Mathematical Formulation

The current iBudget algorithm (Model 5b) employs a multiple linear regression model with square-root transformation:

5 2

p*Yi* = *β*0 + p *βLive · Liveij* + p *βAge · Ageik* + p *βQSI · QSIil* + *εi* (3.14)

where:

*j*

*j*=1

*k l*

*k*=1 *l*

* *Yi* represents FY 2013-14 expenditures for individual *i*
* *Liveij* are dummy variables for living settings (Family Home, ILSL, RH1-RH4)
* *Ageik* are age category indicators (21-30, 31+)
* *QSIil* are Questionnaire for Situational Information scores
* *εi ∼ N* (0*, σ*2) are error terms

The final budget allocation is computed as:

p ˆ 

2



*Budgeti* =

*j*

*βj · Xij*

*· ApportionmentFactor* (3.15)

## Model Performance Metrics

The current algorithm achieves:

*R*2 = 0*.*7998 (3.16)

*noutliers* = 2*,* 410 (9.40% of sample) (3.17)

*ntotal* = 25*,* 615 (after outlier removal) (3.18)

## Critical Mathematical Limitations

### Outlier Dependency

The model’s performance critically depends on outlier removal:

*reduced*

2

*R*

*full*

= 0*.*7549 *≪ R*2

= 0*.*7998 (3.19)

This indicates the algorithm fails to capture the full distribution of individual needs, partic- ularly for complex cases.

### Temporal Validity Issues

Using data from fiscal year 2013-14 introduces significant temporal bias:

*β*ˆ2025 *̸*= *β*ˆ2013*−*14 (3.20)

The assumption of parameter stability over 11+ years is statistically untenable given:

* + - * + Service cost inflation: ∆*Cost ≈* 30% over period
        + Demographic shifts in disability population
        + Changes in service delivery models

### Transformation Bias

The square-root transformation creates systematic bias:

*E*[*Yi|Xi*] *̸*= *E*[*Y*ˆ 2*|Xi*] (3.21)

*i*

This Jensen’s inequality violation leads to consistent underestimation of high-needs individ- uals.

# Compliance Analysis with House Bill 1103

## Person-Centered Planning Deficiencies

The current algorithm violates HB 1103 person-centered requirements through:

*Utilityi* = *f* (*Needsi, Demographicsi*) *⊅ f* (*Preferencesi, Goalsi, Strengthsi*) (3.22) where the algorithm fails to incorporate individual preferences, goals, and strengths as re- quired by statute.

## Data Currency Violations

HB 1103 requires ”recent expenditure data,” but:

*Age*(*Data*) = 2025 *−* 2014 = 11 years *≫* Acceptable threshold (3.23)

# Proposed Alternative Algorithms

## Enhanced Linear Regression Approaches

* + - 1. **Algorithm A1: Robust Linear Regression Mathematical Formulation:**

*n*

*β*ˆ*robust* = arg min

*β*

p*i*=1

*ρ yi* *i*

*− x β*

*T*

*σ*

(3.24)

where *ρ*(*·*) is a robust loss function (Huber or Tukey bisquare):

( 1 *u*2 if *|u| ≤ c*

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

*ρHuber* (*u*) =

**Python Implementation:**

from sklearn . linear\_model import HuberRegressor from sklearn . preprocessing import Standard Scaler import numpy as np

# Robust regression implementation def robust\_ibudget\_algorithm (X, y):

"""

Implements robust linear regression for iBudget allocation

Args:

X: Feature matrix ( n\_samples , n\_features) y: Target expenditures ( n\_samples ,)

Returns:

Trained robust regression model

"""

scaler = Standard Scaler ()

X\_scaled = scaler. fit\_transform ( X)

2

*c|u| −* 1 *c*2 if *|u| > c*

2

(3.25)

20

# Huber regressor handles outliers without removal

model = HuberRegressor ( epsilon =1.35 , alpha = 0 .0001 ) model. fit( X\_scaled , y)

return model , scaler # Usage example

model , scaler = robust\_ibudget\_algorithm ( qsi\_features , expenditures)

predictions = model. predict( scaler. transform ( new\_features))

21

22

23

24

25

26

27

28

* + - 1. **Algorithm A2: Regularized Regression Mathematical Formulation:**

*β*ˆ*LASSO* = arg min

*p*

 1 p

(*yi − xT β*)2 + *λ* p

*|βj|*

(3.26)

*β*  2*n i*=1

*n*

*i*

*n*

*p*

*j*=1 

*β*ˆ*Ridge* = arg min

 1 p

(*yi − xT β*)2 + *λ* p

2

(3.27)

**Python Implementation:**

from sklearn . linear\_model import LassoCV , RidgeCV , ElasticNetCV

from sklearn . model\_selection import cross\_val\_score

method : ’ lasso ’, ’ ridge ’, or ’ elastic ’

Returns:

Optimized regularized model

"""

if method == ’ lasso ’:

model = Lasso CV ( cv =5 , random\_state =42) elif method == ’ ridge ’:

model = Ridge CV ( cv =5) else : # elastic net

model = ElasticNetCV ( cv =5 , random\_state =42) model. fit(X, y)

# Feature importance for transparency importance = np. abs( model. coef\_)

feature\_importance = dict( zip ( range ( len ( importance )), importance )) return model , feature\_importance

# Implementation with QSI features

*i*

*β*

*j*

1

2

*β*  2*n i*=1

*j*=1 

|  |  |  |
| --- | --- | --- |
| 3 |  | |
| 4 | def | regularized\_ibudget\_algorithm (X, y, method =’ elastic ’): |
| 5 |  | """ |
| 6 |  | Implements regularized regression for iBudget allocation |
| 7 |  |  |
| 8 |  | Args: |
| 9 |  | X: Feature matrix including all QSI variables |
| 10 |  | y: Target expenditures |

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

model , importance = regularized\_ibudget\_algorithm ( qsi\_matrix ,

expenditures )

print( f" Selected features: { np. sum ( model. coef\_ != 0)} out of { len ( model. coef\_)}")

33

## Machine Learning Ensemble Approaches

* + - 1. **Algorithm B1: Random Forest Regression Mathematical Formulation:**

*B*

*f*ˆ (*x*) = 1 p *T* (*x*) (3.28)

*D*

*RF*

*B*

*b*=1

*b*

where each tree *Tb* is trained on bootstrap sample *b* with random feature subset.

**Variance Estimation:**

Var[*f*ˆ

*B*

(*x*)] = 1 p Var[*T* (*x*)] (3.29)

*RF*

**Python Implementation:**

*B*2 *b*

*b*=1

|  |  |  |
| --- | --- | --- |
| 1  2  3 | from sklearn . ensemble import Random ForestRegressor from sklearn . model\_selection import Grid Search CV import pandas as pd | |
| 4 |  |  |
| 5 | def | random\_forest\_ibudget\_algorithm (X, y, person\_centered\_features = None ): |
| 6 |  | """ |
| 7 |  | Implements Random Forest for iBudget with person - centered planning |
| 8 |  |  |
| 9 |  | Args: |
| 10 |  | X: QSI and demographic features |
| 11 |  | y: Target expenditures |
| 12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | person\_centered\_features : Individual goals/ preferences  Returns:  Optimized Random Forest model with feature importance  """  # Combine traditional and person - centered features if person\_centered\_features is not None :  X\_combined = np. hstack ([ X, person\_centered\_features ]) else :  X\_combined = X  # Hyperparameter tuning param\_grid = {  ’ n\_estimators ’: [100 , 200 , 500] ,  ’ max\_depth ’: [10 , 20 , None ],  ’ min\_samples\_split ’: [2 , 5 , 10],  ’ min\_samples\_leaf ’: [1 , 2 , 4]  }  rf = Random ForestRegressor ( random\_state =42)  rf\_tuned = Grid Search CV ( rf , param\_grid , cv =5 , scoring =’ r2 ’, n\_jobs  = -1) | |

where *hm*(*x*) minimizes:

|  |  |  |
| --- | --- | --- |
| 33 rf\_tuned . fit( X\_combined , y)  34  35 # Feature importance analysis  36 importance\_df = pd. Data Frame ({  37 ’ feature ’: range ( X\_combined . shape [1]) ,  38 ’ importance ’: rf\_tuned . best\_estimator\_ . feature\_importances\_  39 }). sort\_values(’ importance ’, ascending = False )  40  41 # Prediction intervals using quantile forests  42 from sklearn . ensemble import Random ForestRegressor  43  44 class Quantile Random Forest :  45 def init ( self , \*\* kwargs):  46 self. rf = Random ForestRegressor (\*\* kwargs)  47  48 def fit( self , X, y):  49 self. rf. fit(X, y)  50  51 def predict\_quantiles ( self , X, quantiles =[0.1 , 0.5 , 0.9]):  52 predictions = []  53 for estimator in self. rf. estimators\_:  54 predictions. append ( estimator. predict( X))  55  56 predictions = np. array ( predictions). T  57 return np. quantile ( predictions , quantiles , axis =1). T  58  59 return rf\_tuned . best\_estimator\_ , importance\_df  60  61 # Usage with prediction intervals  62 rf\_model , feature\_importance = random\_forest\_ibudget\_algorithm (  63 qsi\_features , expenditures , person\_centered\_data  64 )  65  66 # Generate prediction intervals for budget planning  67 quantile\_rf = Quantile Random Forest ( n\_estimators =500 , random\_state =42)  68 quantile\_rf. fit( X\_train , y\_train ) | | |
| 69 budget\_intervals = quantile\_rf. predict\_quantiles ( X\_test , [0.1 , | 0.5 , | 0.9]) |
| **3.4.2.2 Algorithm B2: Gradient Boosting with Custom Objective Mathematical Formulation:** |  |  |
| *Fm*(*x*) = *Fm−*1(*x*) + *γmhm*(*x*) |  | (3.30) |

*n*

*hm* = arg min *L*(*yi, Fm−*1(*xi*) + *h*(*xi*)) (3.31)

p

*h*

*i*=1

**Custom Person-Centered Loss Function:**

*LP C*(*yi, y*ˆ*i*) = *α · LMSE*(*yi, y*ˆ*i*) + *β · LP ersonCentered*(*goalsi, y*ˆ*i*) (3.32)

**Python Implementation:**

1

import xgboost as xgb import lightgbm as lgb

from sklearn . metrics import mean\_squared\_error

class Person Centered GradientBoosting : """

Custom gradient boosting with person - centered objective function """

def init ( self , alpha =0.7 , beta =0.3):

self. alpha = alpha # Weight for prediction accuracy self. beta = beta # Weight for person - centered goals

def person\_centered\_objective ( self , y\_pred , y\_true ): """

Custom objective combining MSE with person - centered goals """

# Standard MSE component

mse\_grad = 2 \* ( y\_pred - y\_true . get\_label ()) mse\_hess = np. ones\_like ( y\_pred ) \* 2

# Person - centered component ( example : goal alignment) goal\_alignment = self. \_calculate\_goal\_alignment ( y\_pred , y\_true ) pc\_grad = self. \_person\_centered\_gradient ( y\_pred , goal\_alignment ) pc\_hess = self. \_person\_centered\_hessian ( y\_pred , goal\_alignment )

# Combined objective

grad = self. alpha \* mse\_grad + self. beta \* pc\_grad hess = self. alpha \* mse\_hess + self. beta \* pc\_hess

return grad , hess

def fit( self , X, y, person\_centered\_goals = None ): """

Fit XGBoost model with custom objective """

dtrain = xgb . DMatrix (X, label=y)

params = {

’ objective ’: self. person\_centered\_objective , ’ eval\_metric ’: ’ rmse ’,

’ max\_depth ’: 6 ,

’ learning\_rate ’: 0.1 ,

’ subsample ’: 0.8 ,

’ colsample\_bytree ’: 0.8

}

self. model = xgb . train ( params , dtrain , num\_boost\_round = 1000 ) return self

def predict( self , X): dtest = xgb . DMatrix ( X)

return self. model. predict( dtest)

# Alternative implementation with LightGBM

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

def lightgbm\_ibudget\_algorithm (X, y, categorical\_features = None ):

"""

LightGBM implementation for iBudget allocation """

train\_data = lgb . Dataset(X, label=y, categorical\_feature = categorical\_features )

params = {

’ objective ’: ’ regression ’, ’ metric ’: ’ rmse ’, ’ boosting\_type ’: ’ gbdt ’, ’ num\_leaves ’: 31 ,

’ learning\_rate ’: 0.05 ,

’ feature\_fraction ’: 0.9 ,

’ bagging\_fraction ’: 0.8 ,

’ bagging\_freq ’: 5 ,

’ verbose ’: 0

}

model = lgb . train ( params , train\_data ,

num\_boost\_round =1000 , valid\_sets =[ train\_data ], early\_stopping\_rounds =100 , verbose\_eval= False

)

return model

# SHAP values for explainability import shap

def explain\_predictions ( model , X\_test): """

Generate SHAP explanations for individual predictions """

explainer = shap . Tree Explainer ( model) shap\_values = explainer. shap\_values( X\_test)

# Individual explanation

shap . plots. waterfall( explainer. expected\_value , shap\_values [0], X\_test

. iloc [0]) return shap\_values

# Usage

pc\_gb = Person Centered GradientBoosting () pc\_gb . fit( X\_train , y\_train , person\_centered\_goals ) predictions = pc\_gb . predict( X\_test)

# Explainability

shap\_values = explain\_predictions ( pc\_gb . model , X\_test)

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

## Hybrid Statistical-Clinical Approaches

* + - 1. **Algorithm C1: Two-Stage Hybrid Model Mathematical Formulation:**

**Stage 1 - Base Statistical Model:**

*Y*ˆ*base,i* = *fstat*(*QSIi, Demographicsi, Livingi*) (3.33)

**Stage 2 - Person-Centered Adjustment:**

*Y*ˆ*f inal,i* = *Y*ˆ*base,i ·* (1 + *δi*) (3.34)

where:

*δi* = *gP C*(*Goalsi, Preferencesi, Strengthsi, Contexti*) (3.35)

**Python Implementation:**

from sklearn . base import BaseEstimator , RegressorMixin import numpy as np

class Two Stage Hybrid Model ( BaseEstimator , RegressorMixin ): """

Two - stage hybrid model combining statistical prediction with person - centered adjustments

"""

def init ( self , base\_estimator =None , pc\_weight =0.2): self. base\_estimator = base\_estimator or Random ForestRegressor () self. pc\_weight = pc\_weight

def fit( self , X\_statistical , y, X\_person\_centered = None ): """

Fit the two - stage model

Args:

X\_statistical : Traditional predictors ( QSI , demographics) y: Target expenditures

X\_person\_centered : Person - centered planning features

"""

# Stage 1: Statistical model self. base\_estimator . fit( X\_statistical , y)

base\_predictions = self. base\_estimator . predict( X\_statistical )

# Stage 2: Person - centered adjustment model if X\_person\_centered is not None :

# Calculate residuals for person - centered modeling residuals = y - base\_predictions relative\_residuals = residuals / base\_predictions

# Fit adjustment model

from sklearn . linear\_model import Ridge self. adjustment\_model = Ridge ( alpha =1.0) self. adjustment\_model . fit( X\_person\_centered ,

relative\_residuals )

self. has\_pc\_features = True

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

|  |  |  |
| --- | --- | --- |
| 39 |  | else : |
| 40 |  | self. has\_pc\_features = False |
| 41 |  |  |
| 42 |  | return self |
| 43 |  |  |
| 44 | def | predict( self , X\_statistical , X\_person\_centered = None ): |
| 45 |  | """ |
| 46 |  | Generate predictions using both stages |
| 47 |  | """ |
| 48 |  | # Stage 1 predictions |
| 49 |  | base\_pred = self. base\_estimator . predict( X\_statistical ) |
| 50 |  |  |
| 51 |  | if self. has\_pc\_features and X\_person\_centered is not None : |

52 # Stage 2 adjustments

53 adjustments = self. adjustment\_model . predict( X\_person\_centered

)

54 final\_pred = base\_pred \* (1 + self. pc\_weight \* adjustments)

55 else :

56 final\_pred = base\_pred

57

58 return np. maximum ( final\_pred , 0) # Ensure non - negative budgets

59

60 def get\_explanation ( self , X\_stat , X\_pc , individual\_idx ):

61 """

62 Provide explanation for individual prediction

63 """

64 base\_pred = self. base\_estimator . predict( X\_stat[ individual\_idx : individual\_idx +1])

65

66 explanation = {

67 ’ base\_allocation ’: base\_pred [0],

68 ’ statistical\_factors ’: self. \_get\_statistical\_explanation ( X\_stat[ individual\_idx ]),

69 }

70

71 if self. has\_pc\_features and X\_pc is not None :

72 pc\_adjustment = self. adjustment\_model . predict( X\_pc[ individual\_idx : individual\_idx +1])

73 explanation [’ person\_centered\_adjustment ’] = pc\_adjustment [0]

74 explanation [’ final\_allocation ’] = base\_pred [0] \* (1 + self. pc\_weight \* pc\_adjustment [0])

75 else :

76 explanation [’ final\_allocation ’] = base\_pred [0]

77

78 return explanation

79

80 # Example usage

81 def implement\_two\_stage\_model ( qsi\_data , expenditures , person\_centered\_data ):

82 """

83 Complete implementation of two - stage model

84 """

85 model = Two Stage Hybrid Model (

86 base\_estimator = Random ForestRegressor ( n\_estimators =200) ,

87 pc\_weight =0.15

88

)

model. fit( qsi\_data , expenditures , person\_centered\_data ) # Generate predictions

predictions = model. predict( qsi\_test , pc\_test)

# Individual explanations explanations = []

for i in range ( len ( predictions)):

exp = model. get\_explanation ( qsi\_test , pc\_test , i) explanations. append ( exp )

return model , predictions , explanations # Usage

two\_stage\_model , preds , explanations = implement\_two\_stage\_model (

qsi\_features , expenditures , person\_centered\_features

)

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

* + - 1. **Algorithm C2: Bayesian Hierarchical Model Mathematical Formulation:**

**Level 1 (Individual):**

**Level 2 (Group):**

**Level 3 (Population):**

*Yij|θj, σ*2 *∼ N* (*XT θj, σ*2) (3.36)

*θj|µ,* Σ *∼ N* (*µ,* Σ) (3.37)

*ij*

*µ ∼ N* (*µ*0*,* Σ0)*,* Σ *∼ IW* (*ν*0*, S*0) (3.38)

**Python Implementation:**

|  |  |  |
| --- | --- | --- |
| 1 | import | pymc3 as pm |
| 2 | import | numpy as np |
| 3 | import | pandas as pd |
| 4 | import | theano . tensor as tt |
| 5 |  |  |

6

def bayesian\_hierarchical\_ibudget\_model ( data , group\_var=’ region ’): """

Bayesian hierarchical model for iBudget allocation

Args:

data : Data Frame with individual - level data

group\_var: Grouping variable ( e. g., region , age\_group )

Returns:

PyMC3 model and trace

"""

# Prepare data

groups = data [ group\_var ]. unique ()

group\_idx = data [ group\_var ]. map ({ g: i for i, g in enumerate ( groups)})

. values

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21 n\_groups = len ( groups)

22 n\_obs = len ( data )

23 n\_features = data . select\_dtypes ( include =[ np. number ]). shape [1] - 1

24

25 with pm. Model () as hierarchical\_model :

26 # Hyperpriors

27 mu\_alpha = pm. Normal(’ mu\_alpha ’, 0 , sigma =100)

28 sigma\_alpha = pm. HalfNormal(’ sigma\_alpha ’, sigma =100)

29

30 mu\_beta = pm. Normal(’ mu\_beta ’, 0 , sigma =100 , shape = n\_features)

31 sigma\_beta = pm. HalfNormal(’ sigma\_beta ’, sigma =100 , shape = n\_features)

32

33 # Group - level parameters

34 alpha = pm. Normal(’ alpha ’, mu= mu\_alpha , sigma = sigma\_alpha , shape = n\_groups)

35 beta = pm. Normal(’ beta ’, mu= mu\_beta , sigma = sigma\_beta , shape =( n\_groups , n\_features))

36

37 # Individual - level likelihood

38 X = pm. Data (’X’, data . select\_dtypes ( include =[ np. number ]). iloc[:,

: -1]. values)

39 y\_obs = pm. Data (’ y\_obs ’, data . iloc[:, -1]. values)

40

41 mu = alpha [ group\_idx ] + pm. math . dot(X, beta [ group\_idx ]. T). diagonal ()

42 sigma = pm. HalfNormal(’ sigma ’, sigma =50)

43

44 likelihood = pm. Normal(’y’, mu=mu , sigma = sigma , observed = y\_obs)

45

46 # Sampling

47 trace = pm. sample (2000 , tune =1000 , cores =4 , return\_inferencedata = True )

48

49 return hierarchical\_model , trace

50

51 # Alternative implementation with Stan ( via Py Stan )

52 def stan\_hierarchical\_model ():

53 """

54 Stan implementation for more complex hierarchical models

55 """

56 stan\_code = """

57 data {

58 int < lower =0 > N; // number of observations

59 int < lower =0 > J; // number of groups

60 int < lower =0 > K; // number of predictors

61 int < lower =1 , upper=J> group [ N]; // group indicator

62 matrix [N, K] X; // predictor matrix

63 vector[ N] y; // outcome

64 }

65

66 parameters {

67 real mu\_alpha ;

68 real < lower =0 > sigma\_alpha ;

|  |  |  |
| --- | --- | --- |
| 69 |  | vector[ K] mu\_beta ; |
| 70 |  | vector < lower =0 >[ K] sigma\_beta ; |
| 71 |  |  |
| 72 |  | vector[ J] alpha ; |
| 73 |  | matrix [J, K] beta ; |
| 74 |  | real < lower =0 > sigma ; |
| 75 | } |  |
| 76 |  |  |
| 77 | model { | |
| 78 |  | // Hyperpriors |
| 79 |  | mu\_alpha ~ normal (0 , 100); |
| 80 |  | sigma\_alpha ~ normal (0 , 50); |
| 81 |  | mu\_beta ~ normal (0 , 10); |
| 82 |  | sigma\_beta ~ normal (0 , 10); |
| 83 |  |  |
| 84 |  | // Group - level priors |
| 85 |  | alpha ~ normal( mu\_alpha , sigma\_alpha ); |
| 86 |  | for ( k in 1: K) { |

ormal( mu\_beta [ k], sigma\_beta [ k]);

|  |  |  |
| --- | --- | --- |
| 87 | beta [, k] ~ | n |
| 88 | } |  |
| 89 |  |  |
| 90 | // Likelihood |  |
| 91 | for ( n in 1: N) | { |

92

93 }

94 }

95 """

96

y[ n] ~ normal( alpha [ group [ n]] + X[ n] \* beta [ group [ n]]’, sigma

);

97 return stan\_code

98

99 # Prediction with uncertainty quantification

100 def bayesian\_predictions\_with\_uncertainty ( model , trace , X\_new , group\_new )

:

101 """

102 Generate predictions with full uncertainty quantification

103 """

104 with model:

105 pm. set\_data ({’X’: X\_new , ’ group\_idx ’: group\_new })

106 posterior\_pred = pm. sample\_posterior\_predictive ( trace , samples

= 1000 )

107

108 # Extract prediction intervals

109 predictions = posterior\_pred [’y’]

110

111 pred\_summary = {

112 ’ mean ’: np. mean ( predictions , axis =0),

113 ’ std ’: np. std ( predictions , axis =0),

114 ’ ci\_lower ’: np. percentile ( predictions , 2.5 , axis =0),

115 ’ ci\_upper ’: np. percentile ( predictions , 97.5 , axis =0)

116 }

117

118 return pred\_summary

119

120 # Usage example

121

hierarchical\_model , trace = bayesian\_hierarchical\_ibudget\_model (

budget\_data )

predictions = bayesian\_predictions\_with\_uncertainty ( hierarchical\_model , trace , X\_test , group\_test)

122

## Person-Centered Optimization Approaches

* + - 1. **Algorithm D1: Multi-Objective Optimization Mathematical Formulation:**

where:

min

**b**

subject to

**F**(**b**) = [*f*1(**b**)*, f*2(**b**)*, f*3(**b**)]*T* (3.39)

*n*

p

*bi ≤ Btotal* (3.40)

*i*=1

*bi ≥ bmin,i ∀i* (3.41)

*gj*(**b**) *≤* 0 *j* = 1*, . . . , m* (3.42)

*n*

p

*f*1(**b**) = (*bi −* ˆ*bi*)2 (prediction accuracy) (3.43)

*i*=1

*f*2(**b**) = *−* p *wgoals · GoalAlignmenti*(*bi*) (person-centered goals) (3.44)

*n*

*i*

*i*=1

*n n*

p p

*f*3(**b**) = *|bi − bj| · Similarityij* (fairness) (3.45)

*i*=1 *j*=1

**Python Implementation:**

from pymoo . algorithms. moo . nsga2 import NSGA2 from pymoo . core . problem import Problem

from pymoo . optimize import minimize import numpy as np

class iBudgetMultiObjective Problem ( Problem ): """

Multi - objective optimization problem for iBudget allocation """

def init ( self , predicted\_budgets , person\_centered\_goals , total\_budget , min\_budgets= None ):

self. predicted\_budgets = predicted\_budgets self. person\_centered\_goals = person\_centered\_goals self. total\_budget = total\_budget self. n\_individuals = len ( predicted\_budgets )

self. min\_budgets = min\_budgets or np. zeros( self. n\_individuals )

super (). init ( n\_var= self. n\_individuals , n\_obj =3 ,

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

|  |  |  |
| --- | --- | --- |
| 22 |  | n\_constr =1 , |
| 23 |  | xl= self. min\_budgets , |
| 24 |  | xu=np. full( self. n\_individuals , self. total\_budget) |
| 25 |  | ) |
| 26 |  |  |
| 27 | def | \_evaluate ( self , X, out , \* args , \*\* kwargs): |
| 28 |  | """ |
| 29 |  | Evaluate the multi - objective functions |
| 30 |  | """ |
| 31 |  | n\_solutions = X. shape [0] |

32 f1 = np. zeros( n\_solutions) # Prediction accuracy

33 f2 = np. zeros( n\_solutions) # Person - centered goal alignment

34 f3 = np. zeros( n\_solutions) # Fairness measure

35 g1 = np. zeros( n\_solutions) # Budget constraint

36

37 for i in range ( n\_solutions):

38 budgets = X[i, :]

39

40 # Objective 1: Minimize prediction error

41 f1 [ i] = np. sum (( budgets - self. predicted\_budgets ) \*\* 2)

42

43 # Objective 2: Maximize person - centered goal alignment ( minimize negative )

44 goal\_alignment = self. \_calculate\_goal\_alignment ( budgets)

45 f2 [ i] = - np. sum ( goal\_alignment )

46

47 # Objective 3: Minimize inequality ( Gini coefficient)

48 f3 [ i] = self. \_gini\_coefficient ( budgets)

49

50 # Constraint: Total budget limit

51 g1 [ i] = np. sum ( budgets) - self. total\_budget

52

53 out[" F"] = np. column\_stack ([ f1 , f2 , f3 ])

54 out[" G"] = g1 . reshape (-1 , 1)

55

56 def \_calculate\_goal\_alignment ( self , budgets):

57 """

58 Calculate alignment between budgets and person - centered goals

59 """

60 alignment = np. zeros( len ( budgets))

61 for i, budget in enumerate ( budgets):

62 # Example : alignment based on service categories funded

63 goals = self. person\_centered\_goals [ i]

|  |  |  |
| --- | --- | --- |
| 64 |  | alignment[ i] = self. \_goal\_budget\_alignment ( budget , goals) |
| 65 |  | return alignment |
| 66 |  |  |
| 67 | def | \_goal\_budget\_alignment ( self , budget , goals): |
| 68 |  | """ |
| 69 |  | Calculate how well budget aligns with individual goals |
| 70 |  | """ |
| 71 |  | # Simplified alignment calculation |
| 72 |  | # In practice , this would involve complex service matching |
| 73 |  | target\_services = goals. get(’ preferred\_services ’, []) |
| 74 |  | budget\_adequacy = min ( budget / goals. get(’ estimated\_need ’, budget  ), 1.0) |

75 service\_availability = len ( target\_services ) / 10.0 # Normalize

76

77 return budget\_adequacy \* service\_availability

78

79 def \_gini\_coefficient ( self , budgets):

80 """

81 Calculate Gini coefficient for fairness assessment

82 """

83 sorted\_budgets = np. sort( budgets)

84 n = len ( sorted\_budgets )

85 cumsum = np. cumsum ( sorted\_budgets )

86 return (2 \* np. sum (( np. arange (1 , n + 1) \* sorted\_budgets ))) / ( n

\* cumsum [ -1]) - ( n + 1) / n

87

88 def solve\_multi\_objective\_ibudget ( predicted\_budgets , person\_centered\_goals , total\_budget):

89 """

90 Solve the multi - objective iBudget optimization problem

91 """

92 problem = iBudgetMultiObjective Problem (

93 predicted\_budgets , person\_centered\_goals , total\_budget

94 )

95

96 algorithm = NSGA2 ( pop\_size =100)

97

98 res = minimize (

99 problem ,

100 algorithm ,

101 (’ n\_gen ’, 200) ,

102 verbose = True

103 )

104

105 # Extract Pareto front solutions

106 pareto\_solutions = res. X

107 pareto\_objectives = res. F

108

109 return pareto\_solutions , pareto\_objectives

110

111 # Goal programming alternative

112 from scipy . optimize import minimize as scipy\_minimize

113

114 def goal\_programming\_ibudget ( predicted\_budgets , goals , weights , total\_budget ):

115 """

116 Goal programming approach for person - centered budget allocation

117 """

118 n = len ( predicted\_budgets )

119

120 def objective ( x):

121 budgets = x[: n]

122 pos\_dev = x[ n :2\* n] # Positive deviations

123 neg\_dev = x[2\* n :3\* n] # Negative deviations

124

125 # Weighted sum of deviations from goals

126 return np. sum ( weights[’ accuracy ’] \* ( pos\_dev + neg\_dev ) +

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 127  128 | |  | weights[’ goals ’] \* neg\_dev +  weights[’ fairness ’] \* pos\_dev ) | |
| 129 | |  |  | |
| 130 | | def | constraints( x): | |
| 131 budgets = x[: n]  132 pos\_dev = x[ n :2\* n]  133 neg\_dev = x[2\* n :3\* n]  134 | | | | |
| 135 | | constraints = [] | |  |
| 136 | |  | |  |
| 137 | | # Budget constraint | |  |
| 138 | | constraints. append ( total\_budget - np. sum ( budgets)) | |  |
| 139 | |  | |  |
| 140 | | # Deviation constraints | |  |
| 141 | | for i in range ( n): | |  |
| 142 | | target = goals[ i][’ target\_budget ’] | |  |
| 143 | | constraints. append ( budgets[ i] - target + neg\_dev [ i]  [ i]) | | - pos\_dev |
| 144 | |  | |  |
| 145 | | return np. array ( constraints) | |  |
| 146 | |  | |  |
| 147 | | # Initial guess | |  |
| 148 | | x0 = np. concatenate ([ | |  |
| 149 | | predicted\_budgets , | |  |
| 150 | | np. zeros( n), # positive deviations | |  |
| 151 | | np. zeros( n) # negative deviations | |  |
| 152 | | ]) | |  |
| 153 | |  | |  |
| 154 | | # Bounds | |  |
| 155 | | bounds = ( | |  |
| 156 | | [(0 , total\_budget) for \_ in range ( n)] + # budgets | |  |
| 157 | | [(0 , None ) for \_ in range (2\* n)] # deviations | |  |
| 158 | | ) | |  |
| 159 | |  | |  |
| 160 | | result = scipy\_minimize ( | |  |
| 161 | | objective , x0 , method =’ SLSQP ’, | |  |
| 162 | | constraints ={’ type ’: ’ eq ’, ’ fun ’: constraints}, | |  |
| 163 | | bounds= bounds | |  |
| 164 | | ) | |  |
| 165 | |  | |  |
| 166 | | return result. x[: n] # Return optimized budgets | |  |
| 167 | |  | |  |
| 168 # Us  169 pare | | age example  to\_solutions , objectives = solve\_multi\_objective\_ibudget ( | |  |
| 170 | predicted\_budgets , person\_centered\_goals , total\_budget\_available | | | |
| 171 | ) | | | |
| 172 |  | | | |
| 173 | # Select preferred solution from Pareto front | | | |

174

optimal\_budgets = pareto\_solutions [0] # or use decision - making criteria

* + - 1. **Algorithm D2: Constrained Optimization with Fairness Mathematical Formulation:**

min

**b**

*n*

*i*=1 *n*

p

p

2

*bi −* ˆ*bi*

+ *λ wi*

*i*=1

*n*

p

*GoalScorei −*

*bi* 2

¯*b*

(3.46)

subject to

*bi* = *Btotal* (3.47)

*i*=1

*bi ≥ bmin,i ∀i* (3.48)

p

1 *b*

*i*

*ng i∈G*

*g*

*≥ α ·* ¯*b ∀g ∈ {demographic groups}* (3.49)

1 1 p *b −* 1 p *bi*1 *≤ ϵ ∀a, b ∈ {groups}* (3.50)

1

*b*

*i*

1

2

3

4

5

*na i∈Aa*

**Python Implementation:**

*nb i∈A* 1

6

from scipy . optimize import minimize import cvxpy as cp

import numpy as np

def constrained\_fair\_ibudget\_allocation ( predicted\_budgets , demographic\_groups ,

person\_centered\_scores , total\_budget ,

fairness\_tolerance =0.1):

"""

Constrained optimization with fairness constraints

Args:

predicted\_budgets : Initial statistical predictions demographic\_groups : Group membership for fairness constraints person\_centered\_scores : Individual person - centered alignment

scores

total\_budget: Total available budget

fairness\_tolerance : Maximum allowed group budget difference

Returns:

Optimized budget allocation

"""

n = len ( predicted\_budgets )

unique\_groups = np. unique ( demographic\_groups ) n\_groups = len ( unique\_groups )

# Decision variable

budgets = cp. Variable (n, pos= True )

# Objective function

prediction\_error = cp. sum\_squares( budgets - predicted\_budgets ) person\_centered\_alignment = cp. sum (

cp. multiply ( person\_centered\_scores ,

cp. square ( budgets - np. mean ( predicted\_budgets )))

)

objective = cp. Minimize (

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37 )

38

prediction\_error + 0.1 \* person\_centered\_alignment

39 # Constraints

40 constraints = []

41

42 # Budget constraint

43 constraints. append ( cp. sum ( budgets) == total\_budget)

44

45 # Minimum budget constraints

46 min\_budgets = 0.1 \* predicted\_budgets # 10% minimum

47 constraints. append ( budgets >= min\_budgets)

48

49 # Fairness constraints between demographic groups

50 group\_means = []

51 for group in unique\_groups :

52 group\_mask = ( demographic\_groups == group )

53 group\_indices = np. where ( group\_mask )[0]

54 group\_mean = cp. sum ( budgets[ group\_indices ]) / np. sum ( group\_mask )

55 group\_means. append ( group\_mean )

56

57 # Pairwise fairness constraints

58 for i in range ( n\_groups):

59 for j in range ( i + 1 , n\_groups):

60 constraints. append (

61 cp. abs( group\_means[ i] - group\_means[ j]) <=

62 fairness\_tolerance \* np. mean ( predicted\_budgets )

63 )

64

65 # Solve optimization problem

66 problem = cp. Problem ( objective , constraints)

67 problem . solve ( solver=cp. OSQP )

68

69 if problem . status == cp. OPTIMAL :

70 return budgets. value

71 else :

72 raise Value Error( f" Optimization failed with status: { problem . status}")

73

74 # Alternative formulation with robust optimization

75 def robust\_fair\_ibudget\_allocation ( predicted\_budgets , uncertainty\_sets ,

76 demographic\_groups , total\_budget):

77 """

78 Robust optimization approach handling prediction uncertainty

79 """

80 n = len ( predicted\_budgets )

81

82 # Decision variables

83 budgets = cp. Variable (n, pos= True )

84 slack\_vars = cp. Variable (n, pos= True ) # For robust constraints

85

86 # Worst - case objective considering uncertainty

87 worst\_case\_error = 0

88 for i in range ( n):

89 # Uncertainty set for individual i ( e. g., confidence interval)

90 uncertainty\_radius = uncertainty\_sets [ i]

91 worst\_case\_error += cp. maximum (

92 cp. square ( budgets[ i] - ( predicted\_budgets [ i] + uncertainty\_radius )),

93 cp. square ( budgets[ i] - ( predicted\_budgets [ i] - uncertainty\_radius ))

94 )

95

96 objective = cp. Minimize ( worst\_case\_error + cp. sum ( slack\_vars))

97

98 # Constraints with robustness

99 constraints = [

100 cp. sum ( budgets) == total\_budget ,

101 budgets >= 0.05 \* total\_budget / n, # Minimum allocation

102 slack\_vars >= 0

103 ]

104

105 # Robust fairness constraints

106 unique\_groups = np. unique ( demographic\_groups )

107 for group in unique\_groups :

108 group\_mask = ( demographic\_groups == group )

109 group\_indices = np. where ( group\_mask )[0]

110

111 # Ensure group gets fair share even under uncertainty

112 group\_min\_share = 0.8 \* np. sum ( predicted\_budgets [ group\_mask ])

113 constraints. append (

114 cp. sum ( budgets[ group\_indices ]) >= group\_min\_share - slack\_vars[ group\_indices [0]]

115

116

117

118

119

120

121

122

)

problem = cp. Problem ( objective , constraints) problem . solve ()

return budgets. value , slack\_vars. value # Fairness auditing function

oups ,

|  |  |
| --- | --- |
| 123 | def audit\_allocation\_fairness ( budgets , demographic\_gr  protected\_attributes ): |
| 124 | """ |
| 125 | Comprehensive fairness audit of budget allocation |
| 126 | """ |
| 127 | fairness\_metrics = {} |
| 128 |  |
| 129 | # Statistical parity |
| 130 | for attr in protected\_attributes : |

131 groups = np. unique ( demographic\_groups [ attr ])

132 group\_means = []

133 for group in groups:

134 mask = ( demographic\_groups [ attr] == group )

135 group\_mean = np. mean ( budgets[ mask ])

136 group\_means. append ( group\_mean )

137

138 fairness\_metrics [ f’{ attr} \_statistical\_parity ’] = {

139 ’ group\_means ’: dict( zip ( groups , group\_means)),

140 ’ max\_difference ’: max( group\_means) - min ( group\_means),

141

’ coefficient\_variation ’: np. std ( group\_means) / np. mean (

group\_means)

}

# Equalized opportunity ( for different need levels) # This would require additional need - level data

return fairness\_metrics # Usage example

optimized\_budgets = constrained\_fair\_ibudget\_allocation (

a , pc\_scores , total\_budget

optimized\_budgets , demographic\_data , [’ age\_group ’, ’ disability\_type ’,

’ region ’]

)

142

143

144

145

146

147

148

149

150

|  |  |
| --- | --- |
| 151 | statistical\_predictions , demographic\_dat |
| 152 | ) |
| 153 |  |
| 154 | # Audit the results |
| 155 | fairness\_audit = audit\_allocation\_fairness ( |

156

157

## Modern Time-Aware Approaches

* + - 1. **Algorithm E1: Dynamic Regression with Time Effects Mathematical Formulation:**

**Time-Varying Coefficient Model:**

*Yit* = *XT βt* + *εit* (3.51)

*it*

where coefficients evolve as:

*βt* = *βt−*1 + *ωt, ωt ∼ N* (0*, Q*) (3.52)

**State-Space Representation:**

*βt* = *Fβt−*1 + *ωt* (State equation) (3.53)

*Yt* = *Htβt* + *εt* (Observation equation) (3.54)

**Python Implementation:**

from statsmodels. tsa. statespace import MLEModel from scipy . linalg import block\_diag

import numpy as np import pandas as pd

class DynamicRegressioniBudget ( MLEModel): """

Dynamic regression model for iBudget allocation with time - varying coefficients

"""

def init ( self , endog , exog , \*\* kwargs): self. k\_exog = exog . shape [1]

# Initialize state space model

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15 super (). init (

16 endog ,

17 k\_states= self. k\_exog ,

18 k\_posdef= self. k\_exog ,

19 \*\* kwargs

20 )

21

22 self. exog = exog

23

24 # Initialize state space matrices

25 self. \_initialize\_state\_space ()

26

27 def \_initialize\_state\_space ( self):

28 """

29 Initialize state space representation

30 """

31 # Transition matrix ( random walk for coefficients)

32 self[’ transition ’] = np. eye( self. k\_exog )

33

34 # Selection matrix

35 self[’ selection ’] = np. eye( self. k\_exog )

36

37 # Initial state covariance

38 self[’ state\_cov ’] = np. eye( self. k\_exog )

39

40 def update ( self , params , \*\* kwargs):

41 """

42 Update state space matrices with current parameters

43 """

44 # Parameter mapping

45 obs\_var = params [0]

46 state\_vars = params [1:1+ self. k\_exog ]

47

48 # Update observation equation

49 self[’ obs\_intercept ’] = 0

50 self[’ design ’] = self. exog

51 self[’ obs\_cov ’] = obs\_var

52

53 # Update state equation

54 self[’ state\_cov ’] = np. diag ( state\_vars)

55

56 @ property

57 def param\_names( self):

58 return [’ obs\_var ’] + [ f’ state\_var\_{ i}’ for i in range ( self. k\_exog

)]

59

60 @ property

61 def start\_params( self):

62 return [1.0] + [0.1] \* self. k\_exog

63

64 def fit\_dynamic\_ibudget\_model ( expenditure\_data , qsi\_features , time\_index )

:

65 """

66 Fit dynamic regression model to iBudget data

67

68 Args:

69 expenditure\_data : Time series of expenditures

70 qsi\_features: QSI features over time

71 time\_index : Time index for observations

72

73 Returns:

74 Fitted model and time - varying coefficients

75 """

76 # Prepare data

77 endog = expenditure\_data . values

78 exog = qsi\_features. values

79

80 # Fit model

81 model = DynamicRegressioniBudget ( endog , exog )

82 results = model. fit()

83

84 # Extract time - varying coefficients

85 states = results. states. filtered

86 time\_varying\_coeffs = pd. Data Frame (

87 states.T,

88 index = time\_index ,

89 columns =[ f’ coeff\_{ i}’ for i in range ( qsi\_features. shape [1])]

90 )

91

92 return results , time\_varying\_coeffs

93

94 # Alternative implementation with rolling regression

95 from sklearn . linear\_model import LinearRegression

96 from sklearn . metrics import mean\_squared\_error

97

98 def rolling\_regression\_ibudget ( data , window\_size =12 , min\_periods =6):

99 """

100 Rolling regression approach for time - adaptive iBudget algorithm

101 """

102 results = []

103

104 for i in range ( min\_periods , len ( data )):

105 start\_idx = max (0 , i - window\_size )

106 end\_idx = i + 1

107

108 # Extract window data

109 window\_data = data . iloc[ start\_idx : end\_idx ]

110 X = window\_data . drop (’ expenditure ’, axis =1)

111 y = window\_data [’ expenditure ’]

112

113 # Fit model on window

114 model = LinearRegression ()

115 model. fit(X, y)

116

117 # Store results

118 result = {

119 ’ date ’: data . index [ i],

120 ’ coefficients ’: model. coef\_ ,

121 ’ intercept ’: model. intercept\_ ,

122 ’ r2 ’: model. score (X, y),

d\_error (y, model. predict( X))

|  |  |
| --- | --- |
| 123 | ’ mse ’: mean\_square |
| 124 | } |
| 125 | results. append ( result) |
| 126 |  |

127

128

129

return pd. Data Frame ( results) # Inflation adjustment mechanism

lation\_rates , base\_year

|  |  |
| --- | --- |
| 130 d | ef adjust\_for\_inflation ( historical\_budgets , inf |
|  | = 2024 ): |
| 131 | """ |
| 132 | Adjust historical budget data for inflation |
| 133 | """ |
| 134 | adjusted\_budgets = historical\_budgets . copy () |
| 135 |  |
| 136 | for year , rate in inflation\_rates . items (): |

137 if year != base\_year:

138 adjustment\_factor = (1 + rate ) \*\* ( base\_year - year)

139 year\_mask = adjusted\_budgets . index . year == year

140 adjusted\_budgets . loc[ year\_mask ] \*= adjustment\_factor

141

142 return adjusted\_budgets

143

144 # Forecasting future budget needs

145 from statsmodels. tsa. arima . model import ARIMA

146

horizon =12):

|  |  |
| --- | --- |
| 147 d | ef forecast\_budget\_trends ( time\_series\_data , |
| 148 | """ |
| 149 | Forecast future budget trends using ARIMA |
| 150 | """ |
| 151 | forecasts = {} |
| 152 |  |
| 153 | for column in time\_series\_data . columns: |

154 # Fit ARIMA model

155 model = ARIMA ( time\_series\_data [ column ], order =(1 , 1 , 1))

156 fitted\_model = model. fit()

157

158 # Generate forecasts

159 forecast = fitted\_model. forecast( steps= horizon )

160 conf\_int = fitted\_model. get\_forecast( steps= horizon ). conf\_int ()

161

162 forecasts[ column ] = {

163 ’ forecast ’: forecast ,

164 ’ lower\_bound ’: conf\_int. iloc[:, 0],

165 ’ upper\_bound ’: conf\_int. iloc[:, 1]

166 }

167

168 return forecasts

169

170 # Usage example

171 dynamic\_model , time\_coeffs = fit\_dynamic\_ibudget\_model (

172 expenditure\_time\_series , qsi\_time\_series , date\_index

173 )

174

175 # Rolling regression for comparison

176 rolling\_results = rolling\_regression\_ibudget ( combined\_time\_series\_data )

177

# Forecast future needs

budget\_forecasts = forecast\_budget\_trends ( historical\_budget\_data )

178

179

* + - 1. **Algorithm E2: Longitudinal Mixed-Effects Model Mathematical Formulation:**

**Mixed-Effects Model:**

*Yij* = *XT β* + *ZT bi* + *εij* (3.55)

*ij ij*

where:

*bi ∼ N* (0*, G*) (Random effects) (3.56)

*εij ∼ N* (0*, Rij*) (Within-individual errors) (3.57)

**Individual Growth Curves:**

*Yij* = *β*0 + *β*1*tij* + *β*2*t*2

*ij*

+ *b*0*i* + *b*1*itij* + *εij* (3.58)

**Python Implementation:**

import statsmodels. api as sm

from statsmodels. regression . mixed\_linear\_model import Mixed LM import numpy as np

import pandas as pd

def longitudinal\_ibudget\_model ( data , individual\_id =’ client\_id ’, time\_var= ’ time ’):

"""

Longitudinal mixed - effects model for iBudget needs prediction

Args:

data : Panel data with repeated observations per individual individual\_id : Column name for individual identifier time\_var: Column name for time variable

Returns:

Fitted mixed - effects model

"""

# Prepare fixed effects design matrix

fixed\_effects = [’ age ’, ’ qsi\_behavioral\_sum ’, ’ qsi\_functional\_sum ’, ’ qsi\_physical\_sum ’, time\_var , f’{ time\_var} \_squared ’]

# Add squared time term data [ f’{ time\_var} \_squared ’] = data [ time\_var] \*\* 2

# Fit mixed - effects model model = Mixed LM (

endog = data [’ expenditure ’], exog = data [ fixed\_effects ], groups= data [ individual\_id ], exog\_re = data [[ time\_var ]] # Random slope for time

)

results = model. fit()

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34 return results

35

36 # Alternative implementation with scikit - learn style

37 from sklearn . base import BaseEstimator , RegressorMixin

38 from sklearn . linear\_model import LinearRegression

39

40 class LongitudinaliBudgetPredictor ( BaseEstimator , RegressorMixin ):

41 """

42 Longitudinal predictor for individual budget trajectories

43 """

44

45 def init ( self , max\_time\_horizon =5):

46 self. max\_time\_horizon = max\_time\_horizon

47 self. individual\_models = {}

48 self. population\_model = None

49

50 def fit( self , X, y, individual\_ids , time\_points):

51 """

52 Fit individual trajectory models

53

54 Args:

55 X: Feature matrix

56 y: Target expenditures

57 individual\_ids : Individual identifiers

58 time\_points: Time points for observations

59 """

60 # Fit population - level model

61 X\_with\_time = np. column\_stack ([ X, time\_points , time\_points \*\*2])

62 self. population\_model = LinearRegression ()

63 self. population\_model . fit( X\_with\_time , y)

64

65 # Fit individual models for those with sufficient data

66 unique\_individuals = np. unique ( individual\_ids )

67

68 for ind\_id in unique\_individuals :

69 mask = individual\_ids == ind\_id

70 if np. sum ( mask ) >= 3: # Need at least 3 observations

71 X\_ind = X[ mask ]

72 y\_ind = y[ mask ]

73 t\_ind = time\_points[ mask ]

74

75 # Individual trajectory model

76 X\_ind\_with\_time = np. column\_stack ([ X\_ind , t\_ind , t\_ind

\*\*2])

77 ind\_model = LinearRegression ()

78 ind\_model. fit( X\_ind\_with\_time , y\_ind )

79

|  |  |  |  |
| --- | --- | --- | --- |
| 80 | self. individual\_models [ ind\_id ] = { | |  |
| 81 |  | ’ model ’: ind\_model , |
| 82 |  | ’ last\_observation\_time ’: np. max( t\_ind ), |
| 83 |  | ’ last\_features ’: X\_ind [-1], |
| 84 |  | ’ trajectory\_slope ’: ind\_model. coef\_ [-2] | # Linear |
|  |  | time coefficient |  |
| 85 | } |  |  |
| 86 |  |  |  |

|  |  |  |
| --- | --- | --- |
| 87 |  | return self |
| 88 |  |  |
| 89 | def | predict\_trajectory ( self , individual\_id , future\_time\_points , |
| 90 |  | latest\_features = None ): |
| 91 |  | """ |
| 92 |  | Predict future trajectory for an individual |
| 93 |  | """ |
| 94 |  | if individual\_id in self. individual\_models : |

95 # Use individual model

96 ind\_info = self. individual\_models [ individual\_id ]

97 model = ind\_info [’ model ’]

98

99 if latest\_features is None :

100 latest\_features = ind\_info [’ last\_features ’]

101

102 # Create prediction matrix

103 n\_points = len ( future\_time\_points )

104 X\_pred = np. tile ( latest\_features , ( n\_points , 1))

105 X\_pred = np. column\_stack ([

106 X\_pred ,

107 future\_time\_points ,

108 future\_time\_points \*\*2

109 ])

110

111 return model. predict( X\_pred )

112 else :

113 # Use population model

114 n\_points = len ( future\_time\_points )

115 if latest\_features is None :

116 # Use population averages

117 latest\_features = np. mean ( self. population\_model . coef\_ [: -2])

118

119 X\_pred = np. tile ( latest\_features , ( n\_points , 1))

120 X\_pred = np. column\_stack ([

121 X\_pred ,

122 future\_time\_points ,

123 future\_time\_points \*\*2

|  |  |  |  |
| --- | --- | --- | --- |
| 124 |  | ]) | |
| 125 |  |  | |
| 126 |  | return self. population\_model . predict( X\_pred ) | |
| 127 |  |  | |
| 128 | def | identify\_high\_risk\_individuals ( self , threshold\_slope =100): | |
| 129 |  | """ | |
| 130 |  | Identify individuals with rapidly increasing needs | |
| 131 |  | """ | |
| 132 |  | high\_risk = [] | |
| 133 |  |  | |
| 134  135 |  | for ind\_id , info in self. individual\_models . items ():  if info [’ trajectory\_slope ’] > threshold\_slope : | |
| 136 | hig | | h\_risk . append ({ |
| 137 |  | | ’ individual\_id ’: ind\_id , |
| 138 |  | | ’ slope ’: info [’ trajectory\_slope ’], |
| 139 |  | | ’ last\_time ’: info [’ last\_observation\_time ’] |
| 140 | }) | |  |

141

142 return sorted ( high\_risk , key = lambda x: x[’ slope ’], reverse = True )

143

144 # Survival analysis for service transitions

145 from lifelines import Cox PHFitter

146

147 def service\_transition\_analysis ( data , duration\_col=’ time\_to\_transition ’,

148 event\_col=’ transitioned ’):

149 """

150 Analyze transitions between service levels using survival analysis

151 """

152 # Prepare data for Cox regression

153 cph = Cox PHFitter ()

154

155 # Fit Cox proportional hazards model

156 cph . fit(

157 data ,

158 duration\_col= duration\_col ,

159 event\_col= event\_col

160 )

161

162 return cph

163

164 # Longitudinal clustering for trajectory identification

165 from sklearn . cluster import KMeans

166 from sklearn . preprocessing import Standard Scaler

167

168 def identify\_trajectory\_patterns ( longitudinal\_data , n\_clusters =5):

169 """

170 Identify common trajectory patterns in budget needs

171 """

172 # Reshape data for clustering ( individuals x time points)

173 pivot\_data = longitudinal\_data . pivot\_table (

174 index =’ client\_id ’,

175 columns=’ time ’,

176 values=’ expenditure ’

177 ). fillna ( method =’ ffill ’). fillna ( method =’ bfill ’)

178

179 # Standardize trajectories

180 scaler = Standard Scaler ()

181 scaled\_trajectories = scaler. fit\_transform ( pivot\_data )

182

183 # Cluster trajectories

184 kmeans = KMeans( n\_clusters= n\_clusters , random\_state =42)

185 trajectory\_clusters = kmeans. fit\_predict( scaled\_trajectories )

186

187 # Analyze cluster characteristics

188 cluster\_profiles = {}

189 for cluster\_id in range ( n\_clusters):

190 mask = trajectory\_clusters == cluster\_id

191 cluster\_data = pivot\_data . iloc[ mask ]

192

193 cluster\_profiles [ cluster\_id ] = {

194 ’ n\_individuals ’: np. sum ( mask ),

195 ’ mean\_trajectory ’: cluster\_data . mean ( axis =0),

196

’ std\_trajectory ’: cluster\_data . std ( axis =0),

’ trend ’: ’ increasing ’ if cluster\_data . iloc[:, -1]. mean () > cluster\_data . iloc[:, 0]. mean () else ’ stable ’

}

return trajectory\_clusters , cluster\_profiles # Usage example

longitudinal\_model = longitudinal\_ibudget\_model ( panel\_data )

# Individual trajectory prediction trajectory\_predictor = LongitudinaliBudgetPredictor ()

trajectory\_predictor . fit( X\_features , expenditures , client\_ids , time\_points)

# Predict future needs

future\_times = np. array ([1 , 2 , 3 , 4 , 5]) # Next 5 time periods individual\_forecast = trajectory\_predictor . predict\_trajectory (’ client\_123

’, future\_times)

# Identify high - risk individuals

high\_risk\_clients = trajectory\_predictor . identify\_high\_risk\_individuals ()

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

## Specialized Needs-Based Approaches

* + - 1. **Algorithm F1: Latent Class Mixture Model Mathematical Formulation:**

**Mixture Model:**

where:

*K*

*f* (*yi|xi,* Θ) = *πkfk*(*yi|xi, θk*) (3.59)

p

*k*=1

*πk* = *P* (Individual *i* belongs to class *k*) (3.60)

*fk*(*yi|xi, θk*) = Class-specific density function (3.61)

**EM Algorithm for Estimation:**

**E-step:**

*γ* =  *πkfk*(*yi|xi, θk*)

(3.62)

*ik* L.*K*

*j*=1

*j*

*j*

*i*

*i*

*j*

**M-step:**

*π f* (*y |x , θ* )

*π*(*new*) = 1 p *γ*

*n*

(3.63)

*k n ik*

*i*=1

*θ*(*new*) = arg max p *γik* log *fk*(*yi|xi, θk*) (3.64)

*n*

*k*

**Python Implementation:**

*θk*

*i*=1

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9 | from sklearn . mixture import Gaussian Mixture  from sklearn . linear\_model import LinearRegression import numpy as np  import pandas as pd  class LatentClassiBudgetModel : """  Latent class mixture model for iBudget allocation """ | |
| 10 |  |  |
| 11 | def | init ( self , n\_classes =4 , max\_iter =100 , random\_state =42): |
| 12 |  | self. n\_classes = n\_classes |
| 13 |  | self. max\_iter = max\_iter |
| 14 |  | self. random\_state = random\_state |
| 15 |  | self. class\_models = {} |
| 16 |  | self. mixture\_model = None |
| 17 |  | self. class\_interpretations = {} |
| 18 |  |  |
| 19 | def | fit( self , X, y, feature\_names = None ): |
| 20 |  | """ |
| 21 |  | Fit latent class mixture model |
| 22 |  |  |
| 23 |  | Args: |
| 24 |  | X: Feature matrix ( QSI scores , demographics) |
| 25 |  | y: Target expenditures |
| 26 |  | feature\_names : Names of features for interpretation |
| 27 |  | """ |
| 28 |  | # Step 1: Initial clustering to identify latent classes |
| 29 |  | initial\_gmm = Gaussian Mixture ( |

30

31

32 )

33

n\_components= self. n\_classes , random\_state = self. random\_state

34 # Use both features and outcomes for clustering

35 clustering\_data = np. column\_stack ([ X, y. reshape (-1 , 1)])

36 class\_assignments = initial\_gmm . fit\_predict( clustering\_data )

37

38 # Step 2: Fit class - specific regression models

39 for k in range ( self. n\_classes):

40 class\_mask = ( class\_assignments == k)

41 X\_class = X[ class\_mask ]

42 y\_class = y[ class\_mask ]

43

44 if len ( y\_class) > 10: # Minimum samples for stable estimation

45 model = LinearRegression ()

46 model. fit( X\_class , y\_class)

47

48 self. class\_models[ k] = {

49 ’ model ’: model ,

50 ’ n\_samples ’: len ( y\_class),

51 ’ mean\_expenditure ’: np. mean ( y\_class),

52 ’ mean\_features ’: np. mean ( X\_class , axis =0)

53 }

54

55 # Step 3: Final mixture model for class assignment

56 self. mixture\_model = Gaussian Mixture (

57 n\_components= len ( self. class\_models),

58 random\_state = self. random\_state

59 )

60 self. mixture\_model . fit( X)

61

62 # Step 4: Interpret classes

63 self. \_interpret\_classes (X, y, feature\_names )

64

65 return self

66

67 def \_interpret\_classes ( self , X, y, feature\_names ):

68 """

69 Generate interpretations for each latent class

70 """

71 if feature\_names is None :

72 feature\_names = [ f’ feature\_{ i}’ for i in range ( X. shape [1])]

73

74 for k, class\_info in self. class\_models. items ():

75 mean\_features = class\_info [’ mean\_features ’]

76 mean\_expenditure = class\_info [’ mean\_expenditure ’]

77

78 # Identify distinguishing features

79 overall\_means = np. mean (X, axis =0)

80 feature\_deviations = mean\_features - overall\_means

81

82 # Find most distinctive features

83 top\_features = np. argsort( np. abs( feature\_deviations ))[ -5:]

84

85 interpretation = {

86 ’ class\_size ’: class\_info [’ n\_samples ’],

87 ’ avg\_expenditure ’: mean\_expenditure ,

88 ’ distinguishing\_features ’: [

89 {

90 ’ feature ’: feature\_names [ i],

91 ’ class\_mean ’: mean\_features [ i],

92 ’ overall\_mean ’: overall\_means [ i],

93 ’ deviation ’: feature\_deviations [ i]

94 }

95 for i in top\_features

96 ]

|  |  |  |
| --- | --- | --- |
| 97 |  | } |
| 98 |  |  |
| 99 |  | self. class\_interpretations [ k] = interpretation |
| 100 |  |  |
| 101 | def | predict( self , X): |
| 102 |  | """ |
| 103 |  | Predict expenditures using mixture of class - specific models |
| 104 |  | """ |
| 105 |  | # Get class probabilities |
| 106 |  | class\_probs = self. mixture\_model . predict\_proba ( X) |
| 107 |  |  |
| 108 |  | predictions = np. zeros( len ( X)) |

|  |  |  |  |
| --- | --- | --- | --- |
| 109 |  | | |
| 110 | for i, x in enumerate ( X): | | |
| 111 | class\_prediction = 0 | | |
| 112 | for k, class\_info in self. class\_models. items (): | | |
| 113 | if k < len ( class\_probs[ i]): | | |
| 114  115 | class\_pred = class\_info [’ model ’]. predict( x. reshape (1 ,  -1))[0]  class\_prediction += class\_probs[ i][ k] \* class\_pred | | |
| 116 |  |  |  |
| 117 |  | predictions[ i] | = class\_prediction |
| 118 |  |  |  |
| 119 |  | return predictions |  |
| 120 |  |  |  |
| 121 | def | assign\_class( self , | X): |
| 122 |  | """ |  |
| 123 |  | Assign individuals | to most likely class |
| 124 |  | """ |  |
| 125 |  | return self. mixture | \_model . predict( X) |
| 126 |  |  |  |
| 127 | def | get\_class\_interpret | ation ( self , class\_id ): |
| 128 |  | """ |  |
| 129 |  | Get human - readable | interpretation of a class |
| 130 |  | """ |  |
| 131 | return self. class\_interpretations . get( class\_id , " Class not found "  ) | | |

132

# Usage example

latent\_class\_model = LatentClassiBudgetModel ( n\_classes =5) latent\_class\_model . fit( qsi\_features , expenditures , qsi\_feature\_names )

# Make predictions

predictions = latent\_class\_model . predict( qsi\_test)

# Assign individuals to classes

class\_assignments = latent\_class\_model . assign\_class( qsi\_test)

# Interpret classes

for class\_id in range (5):

interpretation = latent\_class\_model . get\_class\_interpretation ( class\_id

)

print( f" Class { class\_id }: { interpretation }")

133

134

135

136

137

138

139

140

141

142

143

144

145

146

* + - 1. **Algorithm F2: Support Vector Regression Mathematical Formulation:**

**SVR Optimization Problem:**

min

*i*

*w,b,ξ,ξ∗*

*n*

*w* 2 + *C* (*ξi*

1 *∥ ∥* p

2

*i*=1

+ *ξ∗*) (3.65)

subject to *yi − wT ϕ*(*xi*) *− b ≤ ϵ* + *ξi* (3.66)

*wT ϕ*(*xi*) + *b − yi ≤ ϵ* + *ξ∗*

*i*

(3.67)

*ξi, ξ∗ ≥* 0 (3.68)

*i*

**Dual Formulation:**

*f* (*x*) = p(*αi − α∗*)*K*(*xi, x*) + *b* (3.69)

*i*=1

*n*

*i*

where *K*(*xi, xj*) = *ϕ*(*xi*)*T ϕ*(*xj*) is the kernel function.

**Python Implementation:**

from sklearn . svm import SVR

from sklearn . model\_selection import Grid Search CV from sklearn . preprocessing import Standard Scaler from sklearn . pipeline import Pipeline

import numpy as np

class SVRiBudgetAllocator : """

Support Vector Regression for iBudget allocation """

def init ( self , kernel=’ rbf’, multi\_output= False ): self. kernel = kernel

self. multi\_output = multi\_output self. models = {}

self. scaler = Standard Scaler () self. is\_fitted = False

def fit( self , X, y, service\_categories = None ): """

Fit SVR model( s)

Args:

X: Feature matrix

y: Target expenditures ( total or by category ) service\_categories : If provided , fit separate models for each

category

"""

X\_scaled = self. scaler. fit\_transform ( X)

if self. multi\_output and service\_categories is not None : # Fit separate SVR for each service category unique\_categories = np. unique ( service\_categories )

for category in unique\_categories :

# Parameter grid for optimization param\_grid = {

’C’: [0.1 , 1 , 10 , 100] ,

’ epsilon ’: [0.01 , 0.1 , 0.2] ,

’ gamma ’: [’ scale ’, ’ auto ’, 0.001 , 0.01 , 0.1 , 1]

}

svr = SVR ( kernel= self. kernel) grid\_search = Grid Search CV (

svr , param\_grid , cv =5 , scoring =’ r2 ’, n\_jobs = -1

)

# Extract category - specific targets

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

category\_mask = service\_categories == category

y\_category = y[ category\_mask ] if len ( y. shape ) == 1 else y [:, category ]

grid\_search . fit( X\_scaled , y\_category ) self. models[ category ] = grid\_search . best\_estimator\_

else :

# Single SVR model param\_grid = {

’ svr C ’: [0.1 , 1 , 10 , 100 , 1000] ,

’ svr epsilon ’: [0.01 , 0.1 , 0.2 , 0.5] ,

’ svr gamma ’: [’ scale ’, ’ auto ’, 0.001 , 0.01 , 0.1 , 1]

}

pipeline = Pipeline ([

(’ scaler ’, Standard Scaler ()), (’ svr’, SVR ( kernel= self. kernel))

])

grid\_search = Grid Search CV (

pipeline , param\_grid , cv =5 , scoring =’ r2 ’, n\_jobs = -1

)

grid\_search . fit(X, y)

self. models[’ total ’] = grid\_search . best\_estimator\_

self. is\_fitted = True return self

def predict( self , X): """

Generate predictions """

if not self. is\_fitted :

raise Value Error(" Model must be fitted before prediction ")

if len ( self. models) == 1 and ’ total ’ in self. models: return self. models[’ total ’]. predict( X)

else :

# Multi - output prediction predictions = {}

X\_scaled = self. scaler. transform ( X)

for category , model in self. models. items (): predictions[ category ] = model. predict( X\_scaled )

return predictions # Usage example

svr\_allocator = SVRiBudgetAllocator ( kernel=’ rbf’, multi\_output= True )

svr\_allocator . fit( qsi\_features , expenditures , service\_categories )

# Make predictions

svr\_predictions = svr\_allocator . predict( qsi\_test)

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

# Implementation Framework and Validation

## Model Selection Criteria

For algorithm selection, we propose a comprehensive evaluation framework:

**Performance Metrics:**

*n*

*RMSE* = 

r1t 1 p

*n*

(*yi − y*ˆ*i*)2 (3.70)

*i*=1

1 p 1 *yi*

*n*

*− y*ˆ*i* 1

*MAPE* =

*n*

*− −*

*i*=1 1 *yi*

1 *×* 100% (3.71)

**Fairness Metrics:**

*R*2 = 1 *−*

(1 *R*2)(*n* 1)

(3.72)

*n − p −* 1

*adj*

Statistical Parity = max *|E*[*Y*ˆ *|G* = *g*] *− E*[*Y*ˆ *|G* = *h*]*|* (3.73)

*g,h*

Equalized Opportunity = max *|P* (*Y*ˆ *> t|Y > t, G* = *g*) *− P* (*Y*ˆ *> t|Y > t, G* = *h*)*|* (3.74)

*g,h*

**Person-Centered Compliance Score:**

*n*

*i*

*i*

*i*

*PCC* = 1 p GoalAlignment (*Y*ˆ *, Goals , Preferences* ) (3.75)

## Validation Framework

*n*

*i*=1

*i*

1

from sklearn . model\_selection import Time SeriesSplit , cross\_val\_score from sklearn . metrics import mean\_squared\_error , r2 \_score

import numpy as np

def comprehensive\_algorithm\_validation ( models , X, y, temporal\_data = None ): """

Comprehensive validation framework for iBudget algorithms """

validation\_results = {}

for name , model in models. items (): print( f" Validating { name }...")

# Time series validation if temporal data available if temporal\_data is not None :

tscv = Time SeriesSplit ( n\_splits =5) cv\_scores = []

for train\_idx , test\_idx in tscv . split( X): X\_train , X\_test = X[ train\_idx ], X[ test\_idx ] y\_train , y\_test = y[ train\_idx ], y[ test\_idx ]

model. fit( X\_train , y\_train ) y\_pred = model. predict( X\_test)

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

|  |  |  |  |
| --- | --- | --- | --- |
| 26 |  | cv\_scores. append ({ |  |
| 27 |  | ’ rmse ’: np. sqrt( mean\_squared | \_error ( y\_test , y\_pred )), |
| 28 |  | ’ r2 ’: r2 \_score ( y\_test , y\_pre | d ), |
| 29  30 |  | ’ mape ’: np. mean ( np. abs(( y\_te  100  }) | st - y\_pred ) / y\_test)) \* |
| 31 |  |  |  |
| 32 | validation\_results [ name ] = { | |  |
| 33 |  | ’ temporal\_cv ’: cv\_scores , |  |
| 34 |  | ’ mean\_rmse ’: np. mean ([ s[’ rmse ’] | for s in cv\_scores ]), |
| 35 |  | ’ mean\_r2 ’: np. mean ([ s[’ r2 ’] for | s in cv\_scores ]), |
| 36 |  | ’ mean\_mape ’: np. mean ([ s[’ mape ’] | for s in cv\_scores ]) |
| 37 | } |  |  |
| 38 |  |  |  |
| 39 | else : |  |  |

40 # Standard cross - validation

41 cv\_scores = cross\_val\_score ( model , X, y, cv =5 , scoring =’ r2 ’)

42 validation\_results [ name ] = {

43 ’ cv\_r2 \_mean ’: np. mean ( cv\_scores),

44 ’ cv\_r2 \_std ’: np. std ( cv\_scores)

45 }

46

47 return validation\_results

48

49 def fairness\_audit\_framework ( predictions , demographics , protected\_attributes ):

50 """

51 Comprehensive fairness auditing

52 """

53 fairness\_results = {}

54

55 for attr in protected\_attributes :

56 groups = np. unique ( demographics[ attr ])

57 group\_stats = {}

58

59 for group in groups:

60 mask = demographics[ attr] == group

61 group\_stats[ group ] = {

62 ’ mean\_prediction ’: np. mean ( predictions[ mask ]),

63 ’ std\_prediction ’: np. std ( predictions[ mask ]),

64 ’ n\_individuals ’: np. sum ( mask )

65 }

66

67 # Calculate statistical parity difference

68 group\_means = [ stats[’ mean\_prediction ’] for stats in group\_stats. values ()]

69 statistical\_parity = max( group\_means) - min ( group\_means)

70

|  |  |  |
| --- | --- | --- |
| 71 | fairness\_results [ attr] = { | |
| 72 |  | ’ group\_statistics ’: group\_stats , |
| 73 |  | ’ statistical\_parity\_difference ’: statistical\_parity , |
| 74 |  | ’ coefficient\_of\_variation ’: np. std ( group\_means) / np. mean ( |
|  |  | group\_means) |
| 75 | } |  |
| 76 |  |  |

77

return fairness\_results

# Recommendations and Implementation Roadmap

## Phased Implementation Approach

**Phase 1: Foundation Models**

* Implement Algorithm A1 (Robust Linear Regression)
* Implement Algorithm B1 (Random Forest)
* Establish validation framework
* Compare against current Model 5b

**Phase 2: Advanced Approaches**

* Deploy Algorithm C1 (Two-Stage Hybrid)
* Implement Algorithm D1 (Multi-Objective Optimization)
* Conduct fairness audits
* Pilot with subset of enrollees

**Phase 3: Specialized Models**

* Implement Algorithm E1 (Dynamic Regression)
* Deploy Algorithm F1 (Latent Class Mixture)
* Full system integration
* Policy compliance verification

# Conclusion

The current iBudget algorithm exhibits significant limitations in prediction accuracy, temporal validity, and compliance with person-centered planning requirements mandated by House Bill 1103. The proposed collection of alternative algorithms addresses these deficiencies through:

1. **Enhanced statistical robustness** via outlier-resistant methods
2. **Person-centered integration** through multi-objective optimization
3. **Temporal adaptability** using dynamic regression approaches
4. **Specialized population modeling** via mixture models
5. **Fairness assurance** through constrained optimization
6. **Transparency and explainability** via interpretable ML methods

The mathematical formulations and Python implementations provided offer a comprehensive foundation for developing a next-generation iBudget allocation system that meets both statistical rigor and regulatory compliance requirements. The phased implementation approach ensures systematic validation and stakeholder engagement throughout the transition process.

Key success metrics for the new algorithms should include:

*R*2 *≥* 0*.*85 (vs. current 0.80) (3.76)

Outlier Rate *≤* 2% (vs. current 9.4%) (3.77)

Fairness Score *≥* 0*.*95 (3.78)

Person-Centered Compliance *≥* 0*.*90 (3.79) This comprehensive approach ensures Florida’s iBudget system evolves to better serve in- dividuals with developmental disabilities while maintaining fiscal responsibility and regulatory

compliance.