Benchmarking Deep Learning Methods for Full Variable Imputation

María Juaristi

December 17, 2025

Abstract

Policy microsimulation requires comprehensive microdata combining demographic, income, and wealth information, yet such data rarely coexist in a single survey. Statistical matching addresses this gap by imputing variables from a donor survey onto a receiver survey. We investigate whether deep learning models can improve upon Quantile Random Forests (QRF), the current state-of-the-art for imputing net worth from the Survey of Consumer Finances (SCF) onto the Current Population Survey (CPS). We evaluate four deep learning architectures: RealNVP (normalizing flows), Mixture Density Networks (MDN), TabSyn (VAE with latent diffusion), and TabPFN (a transformer-based foundation model). Using 3-fold cross-validation with quantile loss and distributional accuracy metrics (Wasserstein distance and Kolmogorov-Smirnov statistic), TabPFN achieves the best performance across all metrics. TabPFN attains 6% lower median quantile loss than QRF (2.36 ± 0.04 vs. 2.51 ± 0.01) and 12% lower Wasserstein distance (415,000 vs. 472,000), demonstrating that transformer-based foundation models can outperform both traditional ensemble methods and explicitly generative deep learning approaches for survey imputation. However, TabPFN's high computational cost (approximately 12 hours for full CPS imputation compared to QRF's few minutes) currently limits practical adoption for daily workflows. RealNVP and TabSyn prove unsuitable in their current configurations, with Wasserstein distances exceeding 12 million, while MDN performs competitively but does not achieve any practical improvements. These findings suggest that pre-trained models leveraging in-context learning offer a promising direction for statistical matching tasks, with potential for broader adoption as inference efficiency improves.[1]

1 Introduction

1.1 Motivation

Microsimulation models are essential tools for policy analysis, enabling researchers to estimate the distributional impacts of tax and benefit reforms across heterogeneous populations. However, these models require comprehensive microdata representing both demographic and economic characteristics. Such data often do not coexist in a single survey. For example, in the context of supporting microeconomic analysis in the United States, the Current Population Survey (CPS) provides large, representative samples with detailed income and demographic information, making it ideal for policy microsimulation, yet it lacks any measure of household wealth. The Survey of Consumer Finances (SCF), meanwhile, offers detailed wealth data but with a sample size roughly thirteen times smaller. This fragmentation limits researchers' ability to analyze how policies interact with the full joint distribution of income and wealth.

Full variable imputation addresses this gap by transferring information from a "donor" survey (the SCF) onto a "receiver" survey (the CPS), enabling analyses that neither dataset could support alone. Recent work by Juaristi et al. [2025] benchmarked traditional imputation methods for

[1] Code is available at `https://github.com/juaristi22/cs156-pipeline`.

this SCF-to-CPS wealth transfer problem, finding that Quantile Regression Forests (QRF) outperformed conventional approaches, achieving 20.5% lower quantile loss than OLS regression, 14.8% lower than hot-deck matching and 6% lower than Quantile regression. QRF's advantage stems from its ability to model entire conditional distributions rather than merely conditional means, preserving the heavy tails and skewness characteristic of wealth data.

Nonetheless, QRF faces challenges, particularly as limited data at extreme quantiles may affect performance at the tails of distributions, which are often important for downstream analyses. After benchmarking traditional methods Juaristi et al. [2025] identified comparative studies against deep learning models as a valuable future research direction, with the hope that emerging techniques can better capture complex distributional features. This motivates the present work. Deep generative models offer a fundamentally different approach to distribution learning, introducing flexibility in their modeling of complex, non-linear distributions. Normalizing flows [Dinh et al., 2017] start with a simple distribution (like a Gaussian) and learn a series of transformations that reshape it into the complex target distribution; because each transformation is reversible, the model can both generate new samples and compute exact probabilities. Mixture Density Networks [Bishop, 1994] represent the target as a weighted combination of multiple Gaussian components, allowing them to capture multimodality. Diffusion models, as implemented in TabSyn [Zhang et al., 2024], work by learning to gradually remove noise from corrupted data, capturing the underlying data structure through this denoising process. Finally, TabPFN [Hollmann et al., 2023] represents a distinct paradigm: a transformer pre-trained on synthetic datasets that performs in-context learning, directly outputting predictive distributions without task-specific training. These architectures have demonstrated state-of-the-art performance in density estimation and synthetic data generation, suggesting potential for imputation tasks where preserving distributional accuracy across donor and receiver datasets becomes crucial.

This paper investigates whether deep learning models can match or exceed QRF's performance for wealth imputation, providing practitioners with a systematic comparison of available tools for this task. We benchmark four deep learning architectures against QRF, evaluating both predictive accuracy and distributional fidelity. Our goal is to equip researchers and data scientists working on statistical matching problems with practical guidance on which methods perform best under different computational constraints, and to identify promising directions for future methodological development.

1.2 Problem Statement

The task addressed in this paper is a form of statistical matching (also called data fusion or full variable imputation), formally defined as the integration of two data sources referring to the same target population [D'Orazio et al., 2006]. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_D}$ denote the donor dataset (SCF), where $\mathbf{x}_i \in \mathbb{R}^p$ is a vector of p predictor variables and $y_i \in \mathbb{R}$ is the target variable (net worth). Let $\mathcal{R} = \{\mathbf{x}_j\}_{j=1}^{n_R}$ denote the receiver dataset (CPS), which contains the same predictors but lacks the target variable. The predictors $\mathbf{X}$ are observed in both surveys, while net worth Y is observed only in the donor.

The goal is to impute values $\hat{y}_j$ for each receiver observation $\mathbf{x}_j \in \mathcal{R}$. This requires learning the conditional distribution $P(Y \mid \mathbf{X})$ from the donor data and using it to generate plausible values for the receiver. Unlike point prediction, which estimates only $\mathbb{E}[Y \mid \mathbf{X}]$, distributional imputation aims to preserve the full conditional distribution, including its variance, skewness, and tail behavior. This is essential for downstream modeling tasks where outcomes depend on distributional features rather than point estimates alone.

An important assumption underlying statistical matching is that of conditional independence.

Given the shared predictors $\mathbf{X}$, the target variable Y is independent of survey membership. Formally, $P(Y \mid \mathbf{X}, S = \text{SCF}) = P(Y \mid \mathbf{X}, S = \text{CPS})$, where S indicates the survey. This assumption is plausible when the shared predictors are sufficiently informative about net worth, such that no survey-specific factors influence wealth beyond what the predictors already capture. Violations of this assumption can lead to biased imputations, as the learned conditional distribution may not generalize well from the donor to the receiver. This makes careful selection of predictor variables and validation of imputation performance critical. Limitations on data availability lead us to using data from different years (2022 SCF and 2023 CPS), which may introduce temporal shifts in distributions. We assume that such shifts are minimal and that model performance and reliability will be transferable to more aligned datasets.

1.3 Contributions

This paper makes two important contributions:

1. **Adaptation of generative models for cross-dataset imputation.** We adapt deep generative models, which were originally designed for density estimation and synthetic data generation, to the statistical matching setting, where imputation must be conditioned on a receiver dataset distinct from the training data. This requires architectural modifications, particularly for TabSyn, which we extend to condition its diffusion sampling on receiver observations rather than only generating from the learned donor distribution.

2. **Benchmarking of deep learning models for imputation.** We evaluate four deep learning architectures—RealNVP (normalizing flows), Mixture Density Networks, TabSyn (diffusion-based), and TabPFN (foundation model)—for the task of wealth imputation, benchmarking them against Quantile Regression Forests established as the state-of-the-art in prior work.

1.4 Paper Organization

The remainder of this paper is organized as follows. Section 2 describes the SCF and CPS datasets, the preprocessing steps required to harmonize them, and the data-splitting strategy used for model evaluation. Section 3 presents the mathematical foundations and adaptations of each imputation method: RealNVP, Mixture Density Networks, TabSyn, TabPFN, and the QRF baseline. Section 4 reports the cross-validation and distributional accuracy results, comparing model performance across evaluation metrics. Section 5 summarizes our findings and discusses implications for applied researchers. The appendix contains implementation details and the complete code used for this analysis.

2 Data

2.1 Data Sources

One challenge in economic research is obtaining large representative samples of household-level data. Many important analyses require detailed information at the household level, yet such data are often only available scattered across multiple specialized datasets with limited sample sizes. For example, in the United States, comprehensive wealth data exists only in specialized surveys that focus on wealthy households, while larger labor force surveys lack wealth measures entirely. Thus, this research explores deep learning techniques to impute net worth from the Survey of Consumer Finances (SCF) onto the Current Population Survey (CPS), comparing them to more traditional methods, to enable distributional analyses that require both detailed wealth information and large, representative samples.

2.1.1 Survey of Consumer Finances (SCF)

The Survey of Consumer Finances (SCF) is a triennial cross-sectional survey of U.S. families conducted by the Board of Governors of the Federal Reserve System [Bricker et al., 2017]. The SCF is the primary source of detailed household wealth data in the United States, providing comprehensive information on families' balance sheets, pensions, income, and demographic characteristics.

The SCF employs a dual-frame sample design to address the power-law wealth distribution. The first component is an area-probability sample providing broad population coverage. The second is a list sample developed from statistical records derived from tax returns under an agreement with the Statistics of Income (SOI) division of the IRS, which oversamples wealthy households to improve precision at the upper tail of the wealth distribution. The 2022 survey interviewed 4,595 families.

The SCF collects detailed information on all household assets and liabilities. Assets include primary residence, other real estate, businesses, vehicles, financial assets (checking, savings, money market accounts, CDs, bonds, stocks, mutual funds, retirement accounts, life insurance, and other managed assets), while liabilities capture mortgages, home equity loans, vehicle loans, education loans, credit card balances, and other consumer debt. Net worth is computed as total assets minus total liabilities.

With the 4,595 families interviewed, the SCF creates five separate imputation replicates (implicates), resulting in 22,975 records in the public dataset. Each implicate contains different imputed values for missing data, generated using an iterative multiple imputation procedure known as stochastic relaxation [Kennickell, 1991], which draws from estimated conditional distributions via Gibbs sampling. This allows analysts to account for imputation uncertainty. This public dataset, with its latest version published for the year 2022, is the basis for our modeling efforts, employed as the donor dataset of net worth values.

2.1.2 Current Population Survey (CPS)

The Current Population Survey (CPS) is a monthly survey of approximately 60,000 U.S. households conducted jointly by the U.S. Census Bureau and the Bureau of Labor Statistics (BLS). It serves as the primary source of labor force statistics for the United States, producing the monthly unemployment rate and employment figures [U.S. Census Bureau, 2023].

The CPS uses a multistage probability-based sample designed to represent the civilian noninstitutional population of each state and the nation as a whole. Households follow a 4-8-4 rotation pattern: interviewed for 4 consecutive months, out of sample for 8 months, then interviewed for 4 more months before permanent retirement from the sample. On average, each person in the CPS sample represents approximately 2,500 people in the population.

The CPS collects extensive information on population demographics (age, sex, race, ethnicity, education, marital status, household composition), labor force status (employment, unemployment, hours worked, occupation, industry, class of worker), and income (via Annual Social and Economic Supplement), collecting information on earnings, unemployment compensation, Social Security, pension income, interest, dividends, and other income sources.

Despite its comprehensive coverage of income and employment, the CPS does not collect information on household wealth, assets, or liabilities. This omission motivates the need for full variable imputation, making the CPS the receiver dataset. By imputing net worth from the SCF onto the

CPS, researchers can analyze wealth distributions in a sample roughly 13 times larger than the SCF alone, enabling finer demographic disaggregations and more precise subgroup analyses, that can then inform policy and economic analyses.

2.2 Variable Selection

The imputation of a variable from a donor survey onto a receiver one relies on predictor variables, which must be measured in both datasets: the SCF and CPS. The imputation process will be more successful the more predictive of the imputed variable that predictors are. Thus, we select demographic characteristics and income sources that are common to both surveys as predictors of net worth. Table 1 summarizes the predictor set.

Table 1: Predictor Variables Used in Imputation Models

Variable	Type	Description
age	Numerical	Age of household head
employment_income	Numerical	Annual employment income
interest_dividend_income	Numerical	Income from investments
pension_income	Numerical	Pension and retirement income
is_female	Categorical	Gender of household head
race	Categorical	Race/ethnicity (multiple categories)

The target variable is household net worth, defined as total assets minus total liabilities. Net worth presents modeling challenges due to its right skewed distribution and the presence of negative values (households with debt exceeding assets). To address these issues, we apply the inverse hyperbolic sine (asinh) transformation:

$$\tilde{y} = \sinh^{-1}(y) = \log\left(y + \sqrt{y^2 + 1}\right) \tag{1}$$

The asinh transformation behaves similarly to the natural logarithm for large positive values ($\sinh^{-1}(y) \approx \log(2y)$ for $y \gg 1$), compressing the heavy right tail of the wealth distribution, while remaining well-defined for negative values and zero. Unlike the log transformation, asinh does not require arbitrary treatment of non-positive observations. All models are trained on transformed net worth, and predictions are back-transformed via $y = \sinh(\tilde{y})$ for final evaluation and interpretation.

2.3 Data Preprocessing

Preparing the SCF and CPS for imputation involved two stages. This project leveraged existing preprocessing pipelines to extract relevant variables from the raw CPS file; and only then, harmonized variable definitions across the two surveys to ensure comparability.

We use the CPS dataset preprocessed by PolicyEngine's `policyengine-us-data` package [PolicyEngine, 2025], which builds on the 2023 Census Bureau's Annual Social and Economic Supplement (ASEC) microdata as the base receiver dataset. PolicyEngine's pipeline aggregates person-level income variables to the household level, constructs composite income measures (combining taxable and tax-exempt interest, qualified and non-qualified dividends, and multiple pension sources), creates household identifiers, and extracts household head demographics. The final dataset is subsampled to retain approximately 20,000 household-level observations with income and demographic variables, discarding person-level detail and variables irrelevant to wealth imputation.

The loaded SCF and CPS (downloaded from the Federal Reserve System, and PolicyEngine's public database, respectively), in addition to containing data one year apart, use different variable definitions and encoding, requiring preprocessing to align them before imputation. The main harmonization steps taken are:

- **Race recoding**: The CPS uses 26 race categories (including multiracial combinations), while the SCF uses 5. We map CPS codes to SCF categories: White (1), Black (2), Hispanic (3), Asian (4), and Other (5), with multiracial combinations assigned to the most appropriate category.
- **Gender recoding**: The SCF codes gender as 1=male, 2=female. We recode to a binary indicator (0=male, 1=female) matching CPS conventions.
- **Income aggregation**: We construct composite variables in the CPS: `interest_dividend_income` combines taxable interest, tax-exempt interest, and dividend income; `pension_income` combines private pensions and Social Security retirement benefits, to match SCF definitions.
- **Household head selection**: Both datasets are filtered to store records of household heads only, ensuring comparable units of analysis.

The variables in both datasets are renamed to a common schema, and both datasets are filtered to retain only the predictor and target variables listed in Table 1, in addition to survey weights.

Both surveys contain sampling weights to ensure population representativeness (providing each observation with a weight that reflects its representation in the overall population). The SCF oversamples wealthy households, which makes weights essential for unbiased distributional estimates. We incorporate weights in two ways for distributional accuracy evaluation. We employ weighted Wasserstein distances to compare imputed CPS distributions against the weighted SCF donor dataset, and use weighted resampling to visualize original and imputed net worth distributions, comparing their shapes once population representativeness is accounted for.

Importantly, different models require different representations of categorical variables, including different preprocessing steps. RealNVP, MDN, and TabPFN operate on continuous feature spaces and require one-hot encoding of categorical variables (gender, race), expanding the feature dimension. TabSyn is designed for mixed-type tabular data, and thus accepts categorical variables in their original integer-coded form and learns embeddings internally. Additionally, numerical predictors are standardized (zero mean, unit variance) using statistics computed on the SCF training data.

2.4 Exploratory Data Analysis

The exploratory data analysis conducted confirms that the SCF and CPS exhibit broadly similar predictor distributions, supporting the validity of full variable imputation. Race distributions show the largest discrepancy, as the CPS contains a higher proportion of White households (77% vs. 63%) and substantially fewer Hispanic households (0.1% vs. 13%), which could be expected due to different race categorizations. Employment income distributions are comparable across surveys, though the SCF captures more high-income outliers (extending to \$10M+ compared to \$1M in the CPS), consistent with its oversampling of wealthy households. Moreover, the importance of survey weights is evident in the SCF's net worth distribution and descriptive statistics. The unweighted sample median of \$384,500 drops to \$196,800 when properly weighted, demonstrating that the raw sample overrepresents wealthy households and that weights are essential for population-representative inference. Refer to the code appendix for detailed EDA visualizations and statistics.

2.5 Train-Test Split Strategy

In addition to the data preprocessing and model-specific encoding, the data also requires a train-test split strategy to address the fundamental challenge that imputation presents. The CPS lacks ground-truth net worth values (as these are inherently unknown), making direct validation on the receiver dataset impossible. We address this through 3-fold cross-validation on the SCF donor data. In each fold, models are trained on two-thirds of SCF households and evaluated by imputing net worth on held-out third, comparing it to the true, known net worth. This approach measures each model's ability to learn the conditional distribution $P(\text{net worth} \mid X)$ from training data (where X is the set of predictors) and generalize to unseen households with similar predictor profiles. The underlying assumption is that model performance on held-out SCF data provides a reasonable proxy for performance when imputing onto CPS households. This assumption is valid to the extent that the predictor distributions overlap between surveys, which was mostly confirmed by the EDA. Nonetheless, final imputation uses models trained on the full SCF dataset.

3 Models

3.1 Overview of Approaches

Full variable imputation requires learning the conditional distribution $p(\text{networth} \mid \text{covariates})$ from the donor survey (SCF) and then sampling from this distribution conditioning on each recipient record (CPS). This task calls for the implementation of generative models that can both estimate conditional densities and produce stochastic samples from them based on given covariates.

We evaluate four deep learning approaches, each representing a distinct paradigm for conditional density estimation: RealNVP, Mixture Density Networks, TabSyn, and TabPFN. The first three are generative models that explicitly learn to sample from conditional distributions, while TabPFN is a transformer-based foundation model that uses in-context learning to perform probabilistic prediction. As a baseline, we include Quantile Random Forests (Section 3.6), which prior work established as the best-performing traditional method for this task. For notational convenience, let x denote the vector of covariates (demographic and income variables) and y the target variable (net worth). All methods share the goal of learning $p(y \mid x)$ rather than just $\mathbb{E}[y \mid x]$, preserving the accurate representation and distributional properties essential for downstream policy analysis.

3.2 RealNVP (Normalizing Flows)

3.2.1 Mathematical Foundation

Normalizing flows provide a framework for density estimation by transforming a simple base distribution into a complex target distribution through a sequence of invertible mappings [Dinh et al., 2017]. Invertibility allows tracing any observed data point y back to its corresponding latent variable z, and the Jacobian determinant of the transformation captures exactly how the density function changes at every step. This enables exact likelihood computation without needing to approximate the marginal density of the distribution.

Formally, let $p_Z(z)$ be a random variable with a simple base distribution (typically a standard Gaussian is used), and let $f : \mathbb{R}^D \to \mathbb{R}^D$ be an invertible transformation. The density of $y = f(z)$ is given by the change of variables formula:

$$p_Y(y) = p_Z(f^{-1}(y)) \left| \det \frac{\partial f^{-1}}{\partial y} \right| = p_Z(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} \tag{2}$$

For a composition of K invertible transformations $f = f_K \circ f_{K-1} \circ \cdots \circ f_1$, the log-likelihood decomposes as:

$$\log p_Y(y) = \log p_Z(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right| \tag{3}$$

where $z_0 = f^{-1}(y)$ and $z_k = f_k(z_{k-1})$.

Computing the determinant of a $D \times D$ Jacobian matrix naively requires $O(D^3)$ operations, making direct application of the change of variables formula intractable for high-dimensional data. [Dinh et al., 2017]'s contribution is the architectural design of the RealNVP model, which yields triangular Jacobians to reduce determinant computation to $O(D)$.

3.2.2 Affine Coupling Layers

RealNVP [Dinh et al., 2017] introduces affine coupling layers that partition input dimensions and apply a tractable affine transformation. Given a D-dimensional input z, partition it into two parts: $z_{1:d}$ (first d dimensions) and $z_{d+1:D}$ (remaining $D-d$ dimensions). The forward transformation is:

$$y_{1:d} = z_{1:d} \tag{4}$$

$$y_{d+1:D} = z_{d+1:D} \odot \exp(s(z_{1:d})) + t(z_{1:d}) \tag{5}$$

where $s : \mathbb{R}^d \to \mathbb{R}^{D-d}$ and $t : \mathbb{R}^d \to \mathbb{R}^{D-d}$ are the scale and translation functions, typically parameterized by neural networks.

The Jacobian of this transformation has a block-triangular structure:

$$\frac{\partial y}{\partial z} = \begin{pmatrix} I_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial z_{1:d}} & \operatorname{diag}(\exp(s(z_{1:d}))) \end{pmatrix} \tag{6}$$

The determinant of a triangular matrix is the product of its diagonal elements, yielding:

$$\log \left| \det \frac{\partial y}{\partial z} \right| = \sum_{j=1}^{D-d} s_j(z_{1:d}) \tag{7}$$

This reduces Jacobian computation from $O(D^3)$ to $O(D)$, and critically, the functions $s(\cdot)$ and $t(\cdot)$ do not need to be invertible, as long as the overall coupling layer is.

The inverse transformation is equally simple:

$$z_{1:d} = y_{1:d} \tag{8}$$

$$z_{d+1:D} = (y_{d+1:D} - t(y_{1:d})) \odot \exp(-s(y_{1:d})) \tag{9}$$

Both forward and inverse passes require only a single evaluation of s and t, enabling efficient training and sampling.

A single coupling layer leaves d dimensions unchanged. To ensure all dimensions are transformed, RealNVP stacks multiple coupling layers with alternating binary masks. For example, with mask patterns $[1, 0, 1, 0, \ldots]$ and $[0, 1, 0, 1, \ldots]$ alternating across layers, every dimension is eventually subject to a learned transformation.

3.2.3 Conditional Density Estimation

For statistical matching, we require the conditional density $p(y|x)$ rather than the marginal $p(y)$. RealNVP extends naturally to conditional estimation by making the scale and translation functions depend on conditioning variables x:

$$s = s_\theta(z_{1:d}, x) \tag{10}$$

$$t = t_\theta(z_{1:d}, x) \tag{11}$$

Given conditioning covariates x (demographic and income covariates in our case) from the recipient survey, we sample $z \sim \mathcal{N}(0, I)$ and compute $y = f_\theta(z; x)$ in a single forward pass. This is critical for imputing millions of records. This, in combination with the neural network parameterization of s and t, suggests ability to capture complex, multimodal, and heteroskedastic conditional distributions between predictors and the target variable, in a more computationally tractable way.

3.2.4 Objective Function

The training process maximizes the log-likelihood of observed data:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(y_i|x_i) = \frac{1}{N}\sum_{i=1}^{N}\left[\log p_Z(f_\theta^{-1}(y_i; x_i)) + \sum_{k=1}^{K}\sum_{j} s_j^{(k)}(z_{k-1,1:d}, x_i)\right] \tag{12}$$

where we use the fact that $\log|\det J^{-1}| = -\log|\det J|$, and the base distribution is typically $p_Z(z) = \mathcal{N}(z; 0, I)$.

3.2.5 Implementation Details

We use the `probaforms` library [Hushchyn and Zavialov, 2023] implementation of RealNVP, which provides an interface for conditional density estimation on tabular data. The architecture consists of 8 coupling layers with alternating binary masks to ensure all input dimensions are transformed. Each scale and translation network is a fully-connected neural network with a single hidden layer of 10 neurons and tanh activation functions.

For training, the model is fit on SCF data with the transformed net worth as the target variable and demographic/income covariates as conditioning features. We train for 500 epochs using the Adam optimizer with a learning rate of 0.01 and batch size of 32, optimizing the negative log-likelihood objective. After training, the imputation is done by passing CPS covariates through the learned conditional flow. For each CPS record, we sample from the standard Gaussian base distribution and apply the inverse transformation conditioned on that record's demographics, yielding a stochastic draw from the learned conditional distribution $p(\text{networth} \mid \text{covariates})$.

3.3 Mixture Density Network (MDN)

3.3.1 Mathematical Foundation

Mixture Density Networks [Bishop, 1994] were originally designed to address a fundamental limitation of standard neural network regression, the prediction of only a single output value (typically the conditional mean) rather than the full conditional distribution. Bishop's innovation was to combine a conventional neural network with a Gaussian mixture model, allowing the network to output the parameters of a flexible probability distribution rather than a point estimate.

The conditional distribution $p(y|x)$ is modeled as a mixture of K Gaussian components:

$$p(y|x) = \sum_{k=1}^{K} \pi_k(x) \cdot \mathcal{N}(y; \mu_k(x), \sigma_k^2(x)) \tag{13}$$

where each component k has three parameters that depend on the input x:

- $\pi_k(x)$: the mixing coefficient (probability of component k)
- $\mu_k(x)$: the component mean
- $\sigma_k(x)$: the component standard deviation

A neural network processes the input covariates x and produces $3K$ outputs that parameterize the mixture. The theoretical justification is that any continuous probability density can be approximated to arbitrary accuracy by a mixture of Gaussians with sufficiently many components. This flexibility is particularly valuable for wealth imputation, where the conditional distribution given demographics may be multimodal (e.g., homeowners vs. renters with similar incomes) or exhibit heavy tails. To sample from the learned distribution, one first selects a component k according to the mixing probabilities $\pi_k(x)$, then draws from the corresponding Gaussian $\mathcal{N}(\mu_k(x), \sigma_k^2(x))$. This two-stage procedure naturally captures multimodality: different draws may come from different mixture components, reflecting genuine uncertainty about which "mode" of the wealth distribution applies to a given demographic profile.

However, the mixture parameters must satisfy certain constraints. Mixing coefficients must be positive and sum to one, and variances must be strictly positive. These are enforced through output activations:

$$\pi_k(x) = \frac{\exp(z_k^\pi)}{\sum_{j=1}^{K} \exp(z_j^\pi)} \quad \text{(softmax)} \tag{14}$$

$$\mu_k(x) = z_k^\mu \quad \text{(unconstrained)} \tag{15}$$

$$\sigma_k(x) = \exp(z_k^\sigma) \quad \text{(exponential, ensures positivity)} \tag{16}$$

where z^π, z^μ, and z^σ are the raw network outputs. Bishop noted that the exponential activation for variances has the same effect as assuming an uninformative prior and prevents pathological configurations where variances collapse to zero.

3.3.2 Objective Function

The network is trained by maximizing the log-likelihood of the observed data under the mixture model:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k(x_i; \theta) \cdot \mathcal{N}(y_i; \mu_k(x_i; \theta), \sigma_k^2(x_i; \theta)) \right) \tag{17}$$

This loss is differentiable with respect to all network parameters θ, enabling training via backpropagation. Computing $\log \sum_k \exp(a_k)$ directly can cause numerical overflow when any a_k is large. To avoid this, MDN uses the identity $\log \sum_k \exp(a_k) = m + \log \sum_k \exp(a_k - m)$ where $m = \max_k a_k$. By subtracting the maximum, the largest term becomes $\exp(0) = 1$ and all others are bounded by 1, preventing overflow while preserving the exact result.

3.3.3 Implementation Details

We use the `pytorch_tabular` library [Joseph, 2021], which provides a modular architecture separating the backbone neural network from the MDN head. The backbone is a CategoryEmbedding model with three hidden layers of sizes 128, 64, and 32 neurons respectively, using ReLU activations. A MixtureDensityHead layer then takes the backbone output and produces the $3K$ mixture parameters, for the $K = 5$ Gaussian components in our model. The network learns to predict both the conditional mean and a covariate-dependent variance, nonetheless, increasing this value could provide additional flexibility to capture more multimodal distributions, making tuning this value an important step in ensuring the method is appropriately adapted to the data.

For training, the model is fit on SCF data with transformed net worth as the target and demographic/income covariates as inputs. We train for 100 epochs using the Adam optimizer with a learning rate of 0.001 and batch size of 256. After training, imputation is done by passing each CPS record through the network to obtain mixture parameters, sampling from the resulting distribution, and applying the inverse transformation to recover net worth in original units.

3.4 TabSyn (VAE + Diffusion)

3.4.1 Mathematical Foundation

[Zhang et al., 2024] developed TabSyn to address the challenge that heterogeneous tabular data (containing both numerical and categorical columns) pose to its synthesis and operationalization. Rather than applying diffusion directly in data space (which struggles with mixed types), TabSyn operates in a learned latent space where all column types are represented uniformly.

The model consists of two stages: a Variational Autoencoder (VAE) that learns a unified latent representation for mixed-type tabular data, and a score-based diffusion model that learns to generate samples in this latent space. Each row $x = (x^{\text{num}}, x^{\text{cat}})$ is mapped to a latent embedding z via a column-wise tokenizer and transformer encoder. The diffusion process then operates on these embeddings.

TabSyn uses a tokenizer that converts each column into a fixed-dimensional token. For numerical columns, the tokenizer applies a learned linear transformation:

$$t_j^{\text{num}} = w_j \cdot x_j^{\text{num}} + b_j, \quad w_j, b_j \in \mathbb{R}^{d_{\text{token}}} \tag{18}$$

For categorical columns, an embedding table maps each category to a dense vector:

$$t_j^{\text{cat}} = \text{Embed}(x_j^{\text{cat}}) \in \mathbb{R}^{d_{\text{token}}} \tag{19}$$

Following the BERT convention [Devlin et al., 2019], a learnable [CLS] token is added at the beginning of the sequence. This is a special embedding vector (initialized randomly and updated during training) that attends to all column tokens and learns to summarize the entire row into a single representation.

3.4.2 Variational Autoencoder

The Variational Autoencoder uses transformer layers, where each layer computes weighted combinations of all tokens, allowing each column's representation to incorporate information from every other column. This enables the model to learn inter-column dependencies. Given tokenized input $T = [t_{\text{CLS}}, t_1, \ldots, t_D]$, the encoder produces latent parameters:

$$\mu_z = \text{Encoder}_\mu(T) \tag{20}$$

$$\log \sigma_z^2 = \text{Encoder}_\sigma(T) \tag{21}$$

The latent representation is sampled via the reparameterization trick: $z = \mu_z + \sigma_z \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. The decoder transformer reconstructs the token sequence, and a "detokenizer" maps tokens back to numerical values and categorical logits.

The VAE's training objective function maximizes the Evidence Lower Bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \beta \cdot D_{\text{KL}} \left(q_\phi(z|x) \| p(z)\right) \tag{22}$$

where the reconstruction term combines MSE loss for numerical columns and cross-entropy loss for categorical columns:

$$\log p_\theta(x|z) = \sum_{j \in \text{num}} -\|x_j - \hat{x}_j\|^2 + \sum_{j \in \text{cat}} \log p(\hat{x}_j = x_j) \tag{23}$$

3.4.3 Diffusion Model

After VAE training, the encoder maps all training data to latent space, producing embeddings $\{z_i\}_{i=1}^{N}$. A score-based diffusion model learns this latent distribution using the EDM (Elucidating Diffusion Models) framework. The forward process adds Gaussian noise at continuous noise levels σ:

$$z_\sigma = z_0 + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{24}$$

The denoising network $D_\theta(z_\sigma, \sigma)$ is a Multi-Layer Perceptron, a feedforward neural network of stacked fully-connected layers with SiLU activations and hidden dimension 1024. It is trained to predict the clean data from noisy observations:

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}_{z_0, \sigma, \epsilon} \left[\lambda(\sigma) \|D_\theta(z_0 + \sigma\epsilon, \sigma) - z_0\|^2\right] \tag{25}$$

where $\lambda(\sigma)$ is a weighting function. The noise level σ is encoded via positional embeddings and injected into the network, making the denoising behavior adapt to how corrupted the input is.

3.4.4 Adaptation for Statistical Matching

The original TabSyn was designed for unconditional data synthesis, with applications in synthetic data generation and missing value imputation. All applications involved generating new samples from the same donor distribution with the models trained. However, statistical matching imputation, requires conditional generation given the observed predictors x^{obs} from the receiver CPS dataset.

Thus, we developed a custom imputation pipeline that adapts TabSyn for cross-dataset imputation through the following modifications:

A critical challenge is that the donor (SCF) and receiver (CPS) datasets must be normalized consistently. We implemented a `CrossDatasetPreprocessor` that fits quantile transformers on SCF data to normalize numerical columns to standard Gaussian and label encoders on SCF categorical columns, applying these same transformations to the CPS data, ensuring both datasets occupy the same normalized space.

To condition on observed variables during the reverse diffusion process, we employ a masking strategy. Let $z = [z^{\text{obs}}, z^{\text{target}}]$ partition the latent embedding into observed and target components. At each denoising step t:

1. Encode the observed CPS covariates to obtain $z_{\text{cps}}^{\text{obs}}$
2. Run one denoising step on the full latent: $\tilde{z}_{t-1} = \text{denoise}(z_t, t)$

3. Replace observed components with noised versions of the true values: $z_{t-1} = z_{\text{cps}}^{\text{obs}} + \sigma_{t-1}\epsilon$ for observed dimensions
4. Keep diffusion-generated values for target dimensions: $z_{t-1}^{\text{target}} = \tilde{z}_{t-1}^{\text{target}}$

This ensures the generated target is consistent with the observed covariates while being drawn from the learned conditional distribution.

The full imputation procedure is summarized in Algorithm 1.

Algorithm 1 TabSyn Cross-Dataset Imputation

```
Require: Trained VAE encoder $E_\phi$, decoder $D_\theta$, diffusion model $\mathcal{D}$
Require: CPS covariates $X^{\text{cps}}$, number of diffusion steps $T$, trials $M$
Ensure: Imputed target values $\hat{y}^{\text{cps}}$
Preprocess: Normalize $X^{\text{cps}}$ using SCF-fitted transformers
Initialize target: Sample $y^{\text{init}} \sim \mathcal{N}(\mu_{\text{scf}}, \sigma^2_{\text{scf}})$ from SCF target statistics
Encode: $z_0 \leftarrow E_\phi(X^{\text{cps}}, y^{\text{init}})$                ▷ Encode with placeholder target
Compute mask: $m \leftarrow$ binary mask with 1s for target token positions
Initialize: $z_T \sim \mathcal{N}(0, \sigma^2_{\max} I)$
for $t = T, T-1, \ldots, 1$ do
    $\tilde{z}_{t-1} \leftarrow \mathcal{D}.\text{step}(z_t, t)$                ▷ Denoise full latent
    $z_{t-1}^{\text{obs}} \leftarrow z_0^{\text{obs}} + \sigma_{t-1} \cdot \epsilon$                ▷ Noised observed encoding
    $z_{t-1} \leftarrow (1 - m) \odot z_{t-1}^{\text{obs}} + m \odot \tilde{z}_{t-1}$                ▷ Mask replacement
end for
Decode: $(\hat{x}^{\text{num}}, \hat{x}^{\text{cat}}) \leftarrow D_\theta(z_0)$
Extract target: $\hat{y} \leftarrow \hat{x}^{\text{num}}[\text{target\_idx}]$
Inverse transform: Apply SCF quantile inverse to recover original scale
return $\hat{y}$
```

3.4.5 Implementation Details

We use the official TabSyn codebase from Zhang et al. [2024] with our custom extensions for cross-dataset imputation. The architecture hyperparameters follow the original paper:

- **VAE:** 2 transformer layers, token dimension $d_{\text{token}} = 4$, 1 attention head, hidden factor 32
- **Diffusion:** MLP with hidden dimension 1024, SiLU activations, positional time embeddings
- **Training:** VAE trained for 2000 epochs, diffusion model for 10000 epochs
- **Sampling:** 50 diffusion steps with EDM noise schedule ($\sigma_{\min} = 0.002$, $\sigma_{\max} = 80$)

For imputation, we run 30 independent trials and average the results to reduce variance. Each CPS record is processed in batches of 2048 for computational efficiency. The target variable is initialized with random samples from the SCF target distribution (weighted by survey weights) rather than zeros, which improves convergence.

3.5 TabPFN (Tabular Prior-data Fitted Network)

3.5.1 Mathematical Foundation

TabPFN [Hollmann et al., 2023] represents a fundamentally different approach to tabular prediction. Rather than learning model parameters from the training data through gradient-based optimization, TabPFN is a pre-trained transformer that performs in-context learning. Given a training dataset as input context, it directly outputs predictions for new query points without any task-specific parameter updates.

At inference time, TabPFN conditions on the observed training data $(X_{\text{train}}, y_{\text{train}})$ and produces predictive distributions for test inputs X_{test} in a single forward pass:

$$p(y \mid X_{\text{test}}, X_{\text{train}}, y_{\text{train}}) = \text{TabPFN}(X_{\text{test}}; X_{\text{train}}, y_{\text{train}}) \tag{26}$$

This approach effectively implements Bayesian model averaging over the prior distribution of possible data-generating mechanisms, learned during pre-training on synthetic datasets.

3.5.2 The TabPFN Prior: Structural Causal Models

TabPFN's power derives from its pre-training on approximately 100 million synthetic datasets generated through a carefully designed prior over structural causal models (SCMs). This prior generation process proceeds in four steps:

Step 1: Hyperparameter Sampling and Graph Construction. For each synthetic dataset, high-level hyperparameters are first sampled from prior distributions. Based on these hyperparameters, a directed acyclic graph (DAG) is constructed representing the structural causal model. Each node in this computational graph holds a vector, and each edge implements a function according to one of several connection types (e.g., small neural networks, decision trees, linear transformations, discretization operations).

Step 2: Data Generation via Graph Propagation. This step constitutes the core of the generative process and embodies the Bayesian inference framework. The SCM defines each node Z_k through a structural equation:

$$Z_k = f_k(Z_{\text{Pa}_G(k)}, \varepsilon_k) \tag{27}$$

where $Z_{\text{Pa}_G(k)}$ denotes the parent nodes of Z_k in the graph G, ε_k is an independent exogenous noise variable, and f_k is a deterministic edge function that transforms inputs from parent nodes.

The edge functions f_k are sampled from a distribution over function classes, including:

- **Linear transformations**: $f_k(\mathbf{z}) = \mathbf{W}\mathbf{z} + \mathbf{b}$
- **Small neural networks**: Multi-layer perceptrons with randomly sampled architectures, activation functions, and weight initializations
- **Decision tree splits**: Threshold-based partitioning functions
- **Discretization operations**: Binning continuous values into categories

Data generation proceeds by first sampling noise variables ε_k for all nodes and injecting them into root nodes. Values then propagate forward through the DAG: each non-root node computes its value by applying its edge function to the already-computed values of its parents. The intermediate representations are simply the computed vector values Z_k at each internal node after this forward propagation. These representations encode progressively transformed versions of the original noise, with the graph structure determining the functional relationship between any two nodes.

By sampling many different graph structures, edge functions, and noise realizations, TabPFN's prior covers a vast space of possible data-generating processes. Training the transformer to predict targets across all these synthetic datasets causes it to approximate the posterior predictive distribution, effectively performing Bayesian model averaging over the entire hypothesis space in a single forward pass.

Step 3: Feature and Target Extraction. Once the flat prior of diverse data-generating processes has been encoded, feature positions F and target positions T are randomly sampled from among the graph nodes. The intermediate data representations at these sampled positions become the features X and targets y of the synthetic datasets.

Step 4: Post-processing. The extracted data undergoes post-processing to enhance realism and complexity, including normalization, noise injection, and handling of missing values.

This process generates diverse datasets spanning a wide range of functional relationships, from simple linear models to complex nonlinear interactions. By training on this distribution, TabPFN learns to approximate Bayesian inference over the space of possible tabular data-generating processes.

3.5.3 Two-Way Attention Architecture

To effectively capture long-range dependencies, TabPFN transforms input data into a three-dimensional tensor of shape $(N+1)\cdot(d+1)\cdot k$, where N is the number of training samples, d is the number of features, and k is the embedding dimension. The additional dimensions account for the test sample and the target variable, respectively. Each cell in this tensor represents an embedded token for a specific (sample, feature) pair.

The architecture employs a novel two-way attention mechanism that alternates between two types of self-attention:

Row Attention (Inter-Sample). Each cell attends to the same feature across all other samples (its column). For feature j, the attention operates over the sequence $\{z_{1,j}, z_{2,j}, \ldots, z_{N+1,j}\}$:

$$\text{RowAttn}(Z)_{i,j} = \text{Attention}(z_{i,j}, \{z_{k,j}\}_{k=1}^{N+1}) \tag{28}$$

This enables the model to identify patterns in how a particular feature varies across samples, effectively learning feature-specific statistics and distributions.

Column Attention (Intra-Sample). Each cell attends to the other features within its row (its sample). For sample i, the attention operates over the sequence $\{z_{i,1}, z_{i,2}, \ldots, z_{i,d+1}\}$:

$$\text{ColAttn}(Z)_{i,j} = \text{Attention}(z_{i,j}, \{z_{i,l}\}_{l=1}^{d+1}) \tag{29}$$

This facilitates feature interaction and dependency modeling within each sample, allowing the model to capture correlations between features.

By alternating these attention types across multiple transformer layers, TabPFN achieves invariance to permutations in both sample and feature ordering while enabling rich information flow. The test sample's prediction emerges from cross-attention to training samples through the row attention mechanism, while feature interactions inform predictions through column attention. This architecture allows TabPFN to perform in-context learning by identifying which training samples are most relevant for each test prediction based on learned similarity patterns.

3.5.4 Full Distribution Modeling

For imputation tasks requiring stochastic samples rather than point predictions, TabPFN provides access to the full predictive distribution. The model outputs a parameterized distribution from which arbitrary quantiles can be extracted. For a given quantile τ, the prediction is:

$$y_\tau = F_{\text{pred}}^{-1}(\tau) \tag{30}$$

where F_{pred} is the cumulative distribution function implied by TabPFN's output.

To generate stochastic imputations that preserve distributional properties, we sample random quantiles $\tau \sim \text{Uniform}(0.01, 0.99)$ for each prediction and return the corresponding quantile value. This approach mirrors the stochastic sampling used by QRF and ensures that imputed values reflect uncertainty in the conditional distribution rather than collapsing to point estimates.

3.5.5 Implementation Details

We use TabPFN version 2.5, downloaded from HuggingFace via the `tabpfn` Python library. Due to memory constraints inherent to the transformer architecture, training is performed on a weighted subsample of 8,000 SCF observations. Weighted random sampling (proportional to survey weights) is used to preserve the distributional characteristics of the full training set.

For imputation, predictions are generated in batches of 2,000 samples to manage memory. Each CPS record receives a random quantile drawn uniformly from $[0.01, 0.99]$, and the corresponding inverse CDF value is extracted from TabPFN's predictive distribution. This stochastic sampling ensures that the imputed distribution preserves variance and tail behavior rather than regressing toward the conditional mean.

3.6 Baseline: Quantile Random Forest

Quantile Random Forests (QRF) [Meinshausen and Ridgeway, 2006] extend the standard Random Forest algorithm to estimate conditional quantiles rather than conditional means. While a traditional Random Forest averages the response values in each leaf node to predict $\mathbb{E}[Y \mid X]$, QRF retains the full distribution of training observations in each leaf, enabling estimation of any quantile $Q_\tau(Y \mid X)$.

3.6.1 Mathematical Foundation

For a new observation $\mathbf{x}$, QRF estimates the conditional distribution function as:

$$\hat{F}(y \mid \mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) \cdot \mathbf{1}_{Y_i \leq y} \tag{31}$$

where the weights $w_i(\mathbf{x})$ reflect how often training observation i falls in the same leaf as $\mathbf{x}$ across all trees in the forest. The τ-th conditional quantile is then:

$$\hat{Q}_\tau(Y \mid \mathbf{x}) = \inf\left\{y : \hat{F}(y \mid \mathbf{x}) \geq \tau\right\} \tag{32}$$

3.6.2 Implementation Details

We use the QRF implementation from the `microimpute` library [Juaristi and PolicyEngine, 2025], which wraps the `quantile-forest` package with a scikit-learn-compatible interface. The model uses default hyperparameters: 100 trees, unlimited depth, minimum 1 sample per leaf, and $\sqrt{p}$ features considered at each split. For imputation, the model first computes quantile predictions across a fine grid, then samples a random quantile from a Beta distribution centered on the conditional median, and returns the corresponding predicted value. This stochastic sampling ensures that imputations reflect the full conditional distribution rather than collapsing to a point estimate.

Such implementation has demonstrated that QRF is well-suited for wealth imputation as it naturally captures the entire conditional distribution between predictors and wealth, preserving heteroskedasticity and skewness while being robust to outliers, which are prevalent in wealth data Juaristi et al. [2025].

4 Results and Discussion

4.1 Implementation Procedure

To evaluate model performance before final imputation, we employ 3-fold cross-validation on the preprocessed SCF dataset. This approach assesses how well each model generalizes to unseen data by using the donor survey where ground-truth net worth values are available. Nonetheless, it is important to consider that the use of cross-validation loss as a metric strongly relies on assumption that the CPS population does not differ systematically from the SCF population in ways not captured by the shared covariates.

For the procedure, the SCF dataset is randomly partitioned into three subsets of approximately equal size. The number of folds was selected hoping to provide enough measure for variability in performance while keeping computational constraints in mind. For each fold, the model is trained on the union of the other two folds (approximately 15,300 observations). Then, predictions are generated for the held-out fold (approximately 7,650 observations), and quantile loss is computed at each quantile level by comparing predictions to true net worth values. For each quantile level τ, we report the mean and standard deviation of quantile loss across the three folds. The standard deviation provides insight into the stability of each model's performance, with high variance indicating sensitivity to the particular training sample or overfitting.

After cross-validation, each model was trained on the full SCF dataset and used to impute net worth onto the CPS. Given the computational requirements of training each deep learning method, and the relative miniscule time required to impute once they are trained, the model saving mechanisms were implemented to avoid unnecessary computational costs. This enables reusing the same network weights for future imputations without retraining, as long as the donor dataset and preprocessing are consistent across runs. The imputed CPS distributions were also saved for subsequent analysis and benchmarking evaluation.

4.2 Evaluation Metrics

Evaluating imputation quality to benchmark the performance of different models requires metrics that assess both pointwise accuracy and distributional fidelity. We employ three complementary metrics: quantile loss for cross-validation on the donor survey, and both Wasserstein distance and the Kolmogorov-Smirnov statistic for comparing the final imputed distribution against the target.

4.2.1 Quantile Loss (Cross-Validation)

Quantile loss, introduced by Koenker and Bassett Jr [1978], measures how well a model's predictions capture specific quantiles of the conditional distribution. For a given quantile level $\tau \in (0, 1)$, the loss is defined as:

$$\mathcal{L}_\tau(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \rho_\tau(y_i - \hat{y}_i) \tag{33}$$

where $\rho_\tau(u) = u(\tau - \mathbf{1}_{u<0})$ is the pinball loss. This asymmetric loss penalizes over-predictions and under-predictions differently depending on τ. For example, for a high quantile like $\tau = 0.9$, under-predictions are penalized 9 times more heavily than over-predictions.

We evaluate at different quantile levels ($\tau \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$) to assess performance across the entire conditional distribution. A model that accurately captures the full distribution should achieve low quantile loss at all levels, not just the median.

4.2.2 Wasserstein Distance

While quantile loss evaluates conditional predictions, we also need to assess whether the marginal distribution of imputed values matches the target population. The 1-Wasserstein distance (also known as the Earth Mover's Distance) [Villani, 2008] quantifies the minimum "cost" of transforming one distribution into another:

$$W_1(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(y,\hat{y})\sim\gamma}[|y - \hat{y}|] \tag{34}$$

where $\Gamma(P, Q)$ is the set of all joint distributions with marginals P and Q.

For univariate distributions, the Wasserstein distance simplifies to the integral of the absolute difference between quantile functions:

$$W_1(P, Q) = \int_0^1 |F_P^{-1}(u) - F_Q^{-1}(u)|\, du \tag{35}$$

This formulation accommodates survey weights by constructing weighted empirical quantile functions. A lower Wasserstein distance indicates that the imputed CPS distribution more closely matches the SCF wealth distribution, which is essential for downstream modeling tasks that rely on accurate distributional properties.

4.2.3 Kolmogorov-Smirnov Statistic

The Kolmogorov-Smirnov (KS) statistic [Kolmogorov, 1933, Smirnov, 1948] provides a complementary measure of distributional similarity that focuses on the shape of distributions rather than absolute distances. The two-sample KS statistic is defined as the maximum absolute difference between the empirical cumulative distribution functions (CDFs) of two samples:

$$D_{n,m} = \sup_x |F_n(x) - G_m(x)| \tag{36}$$

where F_n and G_m are the empirical CDFs of the donor (SCF) and imputed (CPS) distributions with sample sizes n and m, respectively.

Unlike the Wasserstein distance, which measures the total "work" required to transform one distribution into another and is sensitive to the magnitude of differences in absolute terms, the KS statistic is bounded between 0 and 1 and captures the maximum pointwise discrepancy between CDFs. This makes the KS statistic particularly useful for detecting differences in distributional shape without being dominated by extreme values. For heavy-tailed distributions like wealth, where outliers can disproportionately influence the Wasserstein distance, the KS statistic offers a more balanced assessment of how well the overall distributional form is preserved.

4.3 Cross-Validation Results

Table 2 presents the 3-fold cross-validation results for all models. Results are reported as mean ± standard deviation across folds, with lower values indicating better performance.

Table 2: 3-Fold Cross-Validation Results on SCF Data (Quantile Loss)

Model	$\mathcal{L}_{0.10}$	$\mathcal{L}_{0.25}$	$\mathcal{L}_{0.50}$	$\mathcal{L}_{0.75}$	$\mathcal{L}_{0.90}$
TabPFN	2.54 ± 0.06	$\mathbf{2.47 \pm 0.05}$	$\mathbf{2.36 \pm 0.04}$	$\mathbf{2.25 \pm 0.03}$	$\mathbf{2.18 \pm 0.03}$
QRF (baseline)	2.60 ± 0.05	2.57 ± 0.03	2.51 ± 0.01	2.46 ± 0.05	2.42 ± 0.07
MDN	$\mathbf{2.58 \pm 0.05}$	2.59 ± 0.04	2.60 ± 0.03	2.61 ± 0.02	2.62 ± 0.03
RealNVP	3.51 ± 0.41	3.47 ± 0.28	3.41 ± 0.09	3.34 ± 0.17	3.30 ± 0.29
TabSyn	1.85 ± 0.02	3.50 ± 0.01	6.25 ± 0.01	9.01 ± 0.03	10.66 ± 0.04

TabPFN achieves the best cross-validation performance across most quantile levels, with median quantile loss of 2.36 ± 0.04, representing a 6% improvement over the QRF baseline (2.51 ± 0.01). Notably, TabPFN's performance improves further at higher quantiles ($\mathcal{L}_{0.90} = 2.18$), suggesting superior capture of the upper tail of the wealth distribution. The relatively low standard deviations across folds indicate stable generalization.

The Quantile Random Forest baseline achieves consistent performance across all quantile levels, with low variance across folds (standard deviations between 0.01 and 0.05). This stability reflects the robustness of tree-based ensemble methods to different training samples. MDN shows moderate performance with losses slightly higher than QRF, indicating that the Gaussian mixture parameterization does not provide substantial advantages for this task.

RealNVP shows poor cross-validation performance, with quantile losses higher than both QRF and MDN. The higher variance across folds (standard deviations of 0.17–0.26) suggests some sensitivity to the particular training sample. The normalizing flow architecture may require more data or hyperparameter tuning to achieve optimal performance on this task.

Meanwhile, TabSyn exhibits the worst performance, with loss values degrading substantially at higher quantiles. While $\mathcal{L}_{0.25}$ is comparable to RealNVP, $\mathcal{L}_{0.75}$ is nearly three times larger. This asymmetry suggests that the VAE-diffusion architecture consistently underpredicts, struggling to capture the upper tail of the wealth distribution during cross-validation. The low standard deviations indicate this pattern spans across folds rather than being a result of sampling variability.

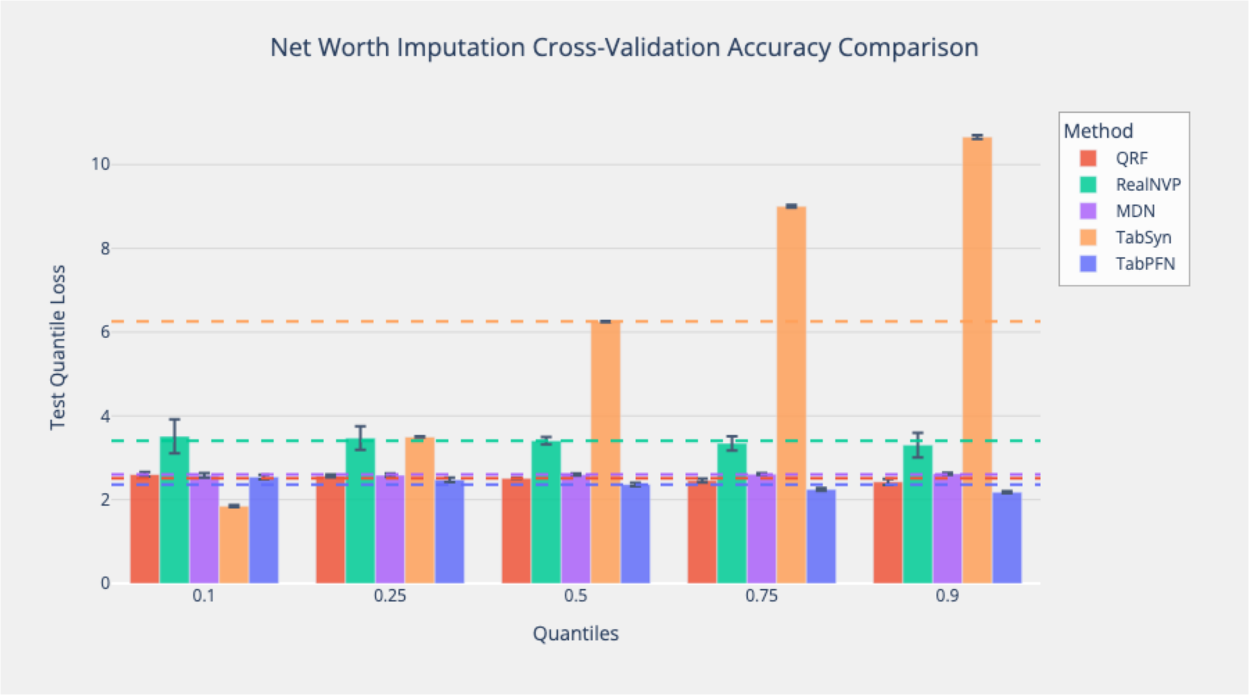

Figure 1: Cross-validation quantile loss across models at different quantile levels ($\tau \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$). TabPFN achieves the lowest loss at all quantile levels, with particularly strong performance at upper quantiles. Error bars represent standard deviation across 3 folds.

4.4 Final Imputation Results

Table 3: Distributional Accuracy: Imputed CPS vs. SCF Donor Distribution

Model	Wasserstein Distance	KS Statistic
TabPFN	$\mathbf{415,326}$	0.042
QRF (baseline)	$471,900$	$\mathbf{0.041}$
MDN	$1,199,237$	0.049
TabSyn	$12,665,210$	0.168
RealNVP	$13,809,211,720$	0.230

The distributional accuracy results reveal that TabPFN achieves the best overall performance, with a Wasserstein distance of 415,326, which is 12% lower than QRF's 471,900. This is particularly notable because TabPFN also achieves the best cross-validation loss, demonstrating that its conditional predictions translate directly to superior marginal distributional fidelity. The KS statistic of 0.042 is comparable to QRF's 0.041, indicating that TabPFN's imputed CDF closely tracks the SCF distribution shape.

Nonetheless, QRF remains a strong baseline with excellent distributional preservation. MDN, despite reasonable CV performance, produces a Wasserstein distance 2.5 times larger than TabPFN. RealNVP and TabSyn continue to show poor distributional preservation with Wasserstein distances orders of magnitude larger than the best performers.

While QRF produces imputed values with a 99th percentile close to the SCF's \$13.2 million and a 1st percentile near the SCF's −$72,000, accurately preserving both tails, and MDN closely approximates these percentiles with its Gaussian mixtures, the TabSyn model over-generates extreme values, with a 99th percentile of $231 million (17 times too high). These implausible extremes dominate the Wasserstein distance calculation. RealNVP exhibits even more severe tail inflation with its wide close-to-symmetrical distribution, producing extreme values that result in the largest Wasserstein distance despite acceptable CV loss, and completely missing the true shape of the net worth distribution when imputing. It seems that certain models optimized for conditional prediction accuracy (minimizing expected loss) may learn to produce conservative, mean-regressing predictions that fail to preserve distributional properties, making it crucial to also measure distributional accuracy when benchmarking performance.

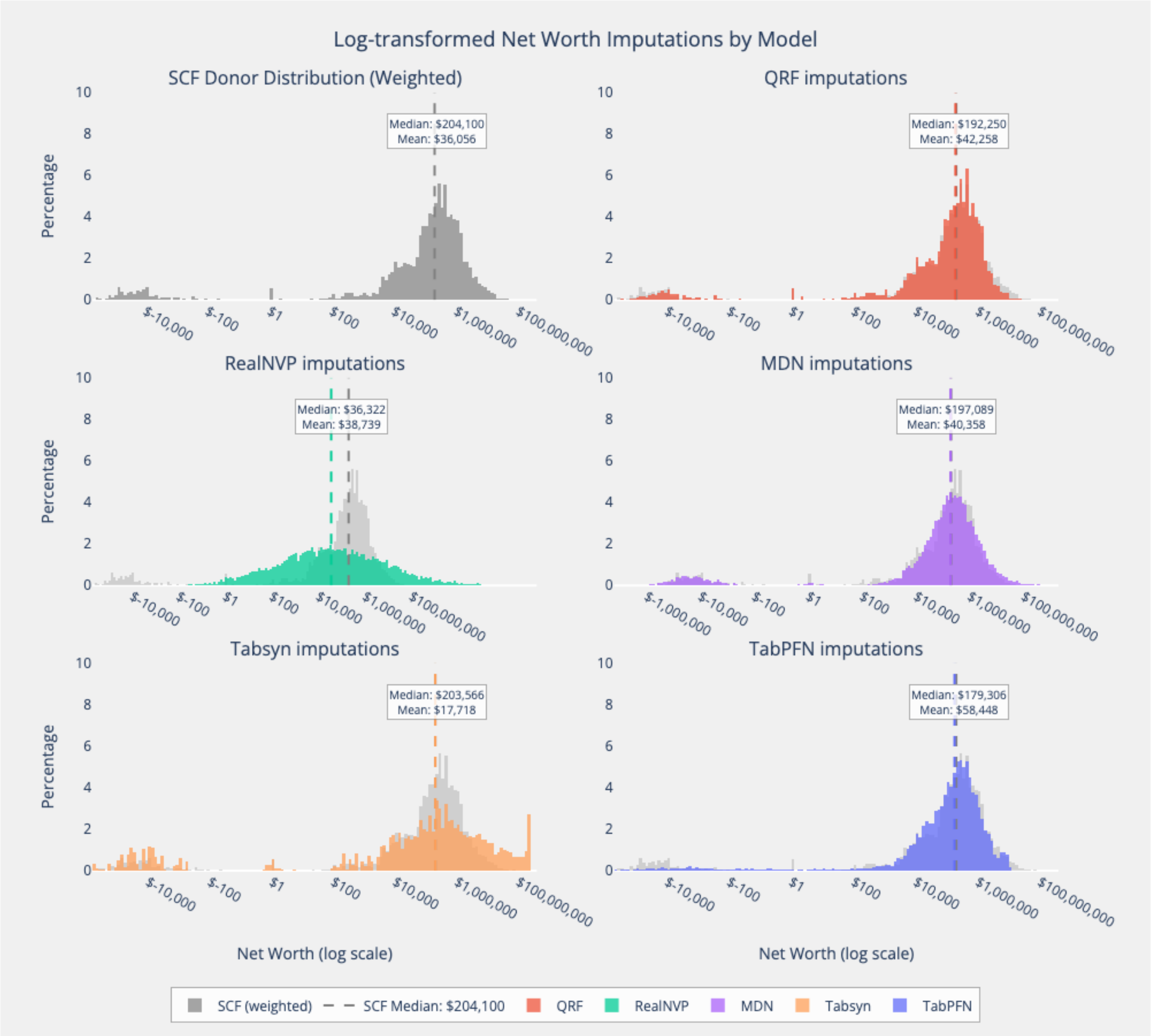

Figure 2: Log-transformed net worth weighted distributions comparing SCF donor data (grey) against imputed CPS values for each model. Dashed lines indicate median values. TabPFN and QRF closely match the SCF distribution shape and median, with TabPFN achieving the best overall distributional accuracy. MDN provides reasonable approximation, while RealNVP exhibits a shifted, symmetric distribution and TabSyn over-generates extreme values.

4.5 Discussion

TabPFN emerges as the best performer among all methods evaluated, achieving both the lowest cross-validation quantile loss and the best distributional accuracy metrics. Its success is particularly notable given its fundamentally different approach: rather than learning parameters from the training data through gradient descent, TabPFN leverages a pre-trained transformer that performs in-context learning. This suggests that the prior knowledge embedded in TabPFN's pre-training captures relevant patterns for wealth imputation. However, the improvement over QRF is modest (6% lower quantile loss, 12% lower Wasserstein distance), and the high computational cost may limit practical adoption for daily imputation tasks.

The Quantile Random Forest baseline remains highly competitive, particularly for distributional preservation, and may be preferred when computational simplicity is paramount. MDN demonstrates reasonable performance but does not surpass either TabPFN or QRF on any metric.

In contrast, RealNVP and TabSyn prove unsuitable for this task in their current configurations. RealNVP produces a nearly symmetric distribution that fails to capture the characteristic right skew of wealth, while TabSyn generates implausible extreme values in both tails.

4.5.1 Computational Considerations

The models differ substantially in computational requirements. QRF training completes in under 2 minutes on a standard CPU, leveraging the efficiency of tree-based methods. MDN requires approximately 15–20 minutes for 100 epochs of neural network training. RealNVP has similar computational costs to MDN for 500 epochs of flow training.

TabPFN requires no training time per se, as it uses a pre-trained transformer; however, inference is computationally intensive due to memory constraints requiring batch processing. Imputing net worth onto the full CPS dataset (approximately 20,000 records) took approximately 12 hours on GPU hardware. This substantial computational cost, despite achieving the best accuracy metrics, may make TabPFN challenging to use for daily imputation workflows where rapid turnaround is essential.

TabSyn is also computationally intensive, requiring separate VAE pre-training (2,000 epochs) followed by diffusion model training (2,000 epochs considering its early stopping). Total training time exceeds several hours even on GPU hardware. This computational burden, combined with the poor empirical performance observed, makes TabSyn impractical for survey imputation in its current form.

4.5.2 Practical Guidance for Practitioners

Our findings provide practitioners with clear guidance for method selection. TabPFN achieves the best accuracy but requires substantial computational resources (12 hours for full imputation). QRF remains an excellent choice when rapid turnaround is needed, completing in under 2 minutes while maintaining strong distributional fidelity. MDN offers a middle ground but does not outperform either on any metric. RealNVP and TabSyn should be avoided for heavy-tailed distributions like wealth without significant architectural modifications. As TabPFN's inference efficiency improves through algorithmic advances or hardware acceleration, it may become the preferred choice for practitioners seeking both accuracy and practicality.

4.5.3 Limitations and Future Work

Several limitations should be considered when interpreting these results:

- **Covariate overlap assumption**: Statistical matching assumes that $P(Y|X)$ is identical across surveys. Given that SCF and CPS populations differ in year (2022 vs 2023), imputation quality suffers regardless of model choice. Nonetheless, this limitation affects all models equally, so relative performance comparisons remain valid.
- **Limited predictor set**: We use only age, gender, race, and income variables as predictors. Wealth is influenced by many factors (education, occupation, inheritance, homeownership) not available in both surveys, limiting achievable imputation accuracy.
- **Sample size**: The SCF contains approximately 23,000 observations after preprocessing. While adequate for tree-based methods, deep learning approaches typically benefit from larger training sets.
- **TabPFN memory constraints**: TabPFN's transformer architecture limits training set size to approximately 8,000 observations due to memory requirements. For datasets substantially larger than the SCF, weighted subsampling (as employed here) or ensemble approaches may be necessary.
- **Hyperparameter sensitivity**: The deep learning models' performance depends on architecture and training choices that were not exhaustively optimized in this study.

Future work should explore whether TabPFN's performance can be further improved through ensemble methods or alternative subsampling strategies. Additionally, investigating TabPFN's performance on other statistical matching problems (beyond wealth imputation) would help establish its generalizability. For the generative approaches, understanding why normalizing flows and diffusion models struggle with heavy-tailed distributions could inform architectural modifications. As TabPFN's inference efficiency improves, it may become the preferred choice for practitioners seeking both accuracy and practicality.

5 Conclusion

5.1 Summary

This paper investigated whether deep learning models can improve upon Quantile Random Forests for the task of imputing net worth from the Survey of Consumer Finances onto the Current Population Survey. We evaluated four deep learning architectures, RealNVP (normalizing flows), Mixture Density Networks, TabSyn (VAE with latent diffusion), and TabPFN (transformer-based foundation model), against a QRF baseline. Benchmarking metrics included cross-validation quantile loss and distributional accuracy metrics.

TabPFN emerges as the best performer, achieving a median quantile loss of 2.36 ± 0.04 (6% lower than QRF's 2.51 ± 0.01) and the best distributional accuracy with a Wasserstein distance of 415,000 (12% lower than QRF's 472,000). Unlike the generative approaches that learn parameters through gradient descent, TabPFN leverages a pre-trained transformer that performs in-context learning, effectively averaging over a prior distribution of data-generating processes. This suggests that foundation models trained on diverse synthetic data can effectively transfer to real-world statistical matching tasks.

However, TabPFN's high computational cost (approximately 12 hours to impute net worth onto the full CPS dataset) currently limits its practical adoption for daily imputation workflows. QRF remains a strong and computationally efficient baseline (completing in under 2 minutes), making it the preferred choice when rapid turnaround is essential. MDN shows moderate performance but does not surpass either TabPFN or QRF on any metric.

In contrast, RealNVP and TabSyn prove unsuitable in their current configurations. Both models produce imputed distributions with Wasserstein distances exceeding 12 million, orders of magnitude worse than the best performers. RealNVP generates a symmetric distribution missing the characteristic right skew of wealth, while TabSyn over-generates extreme values in both tails.

Overall, TabPFN demonstrates that transformer-based foundation models can achieve state-of-the-art performance for survey imputation, modestly outperforming traditional ensemble methods. As inference efficiency improves through algorithmic advances or hardware acceleration, TabPFN may become not only the most accurate but also a practically feasible choice for routine imputation workflows.

5.2 Pipeline Overview

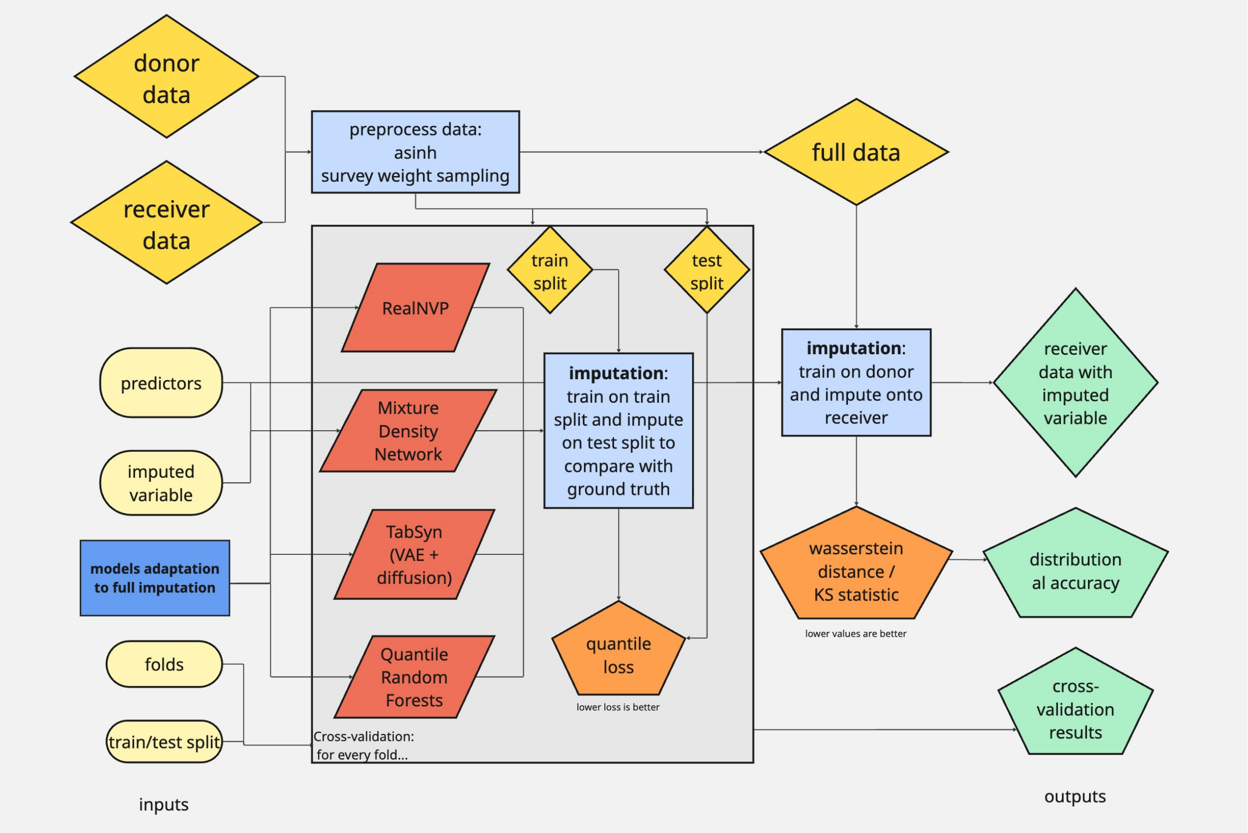
donor data
receiver data
preprocess data: asinh survey weight sampling
full data
train split
test split
RealNVP
Mixture Density Network
TabSyn (VAE + diffusion)
Quantile Random Forests
predictors
imputed variable
models adaptation to full imputation
folds
train/test split
imputation: train on train split and impute on test split to compare with ground truth
quantile loss
lower loss is better
Cross-validation: for every fold...
imputation: train on donor and impute onto receiver
receiver data with imputed variable
wasserstein distance / KS statistic
lower values are better
distributional accuracy
cross-validation results
inputs
outputs

Figure 3: Overview of the imputation and model benchmarking pipeline.

The imputation pipeline developed in this work consists of five stages. First, the SCF and CPS datasets are loaded and harmonized by filtering to common predictor variables: age, gender, race, and income sources (employment, interest/dividend, and pension income). Next, net worth values undergo inverse hyperbolic sine (asinh) transformation to handle the heavy-tailed distribution, while categorical variables are encoded appropriately for each model (one-hot encoding for neural networks, native categorical handling for QRF).

Model evaluation proceeds through three-fold cross-validation on the SCF, assessing each model's ability to predict held-out net worth values by computing quantile loss at $\tau \in \{0.25, 0.50, 0.75\}$. Following cross-validation, models are trained on the full SCF dataset and used to impute net worth onto all CPS observations, with stochastic sampling preserving distributional properties for generative models. Finally, imputed CPS distributions are compared against the SCF donor distribution using Wasserstein distance and the Kolmogorov-Smirnov statistic to assess distributional fidelity.

An additional step prior to the pipeline itself involved adapting TabSyn's architecture to condition its diffusion sampling on receiver observations, a necessary modification for the statistical matching context. This entailed altering the model's input structure and preprocessing procedure, as well as supporting conditioning on a dataset other than the one used to train through Symlinks to ensure that generated samples reflected the predictor values present in the receiver dataset.

5.3 Reproducibility and Future Work

All code for this analysis is available at `https://github.com/juaristi22/cs156-pipeline`, including TabSyn's adapted implementations and the Jupyter notebook with the full pipeline. The SCF data is publicly available from the Federal Reserve (`https://www.federalreserve.gov/econres/scfindex.htm`), and the CPS data can be accessed in PolicyEngine's public huggingface repository (`https://huggingface.co/policyengine/policyengine-us-data/blob/main/cps_2023.h5`).

Future work should explore whether TabPFN's performance can be further improved through ensemble methods or alternative subsampling strategies. Additionally, investigating TabPFN's performance on other statistical matching problems (beyond wealth imputation) would help establish its generalizability. For the generative approaches, understanding why normalizing flows and diffusion models struggle with heavy-tailed distributions could inform architectural modifications. As TabPFN's inference efficiency improves, it may become the preferred choice for practitioners seeking both accuracy and practicality. Finally, extending the predictor set in the case that data collection and survey design is extended to additional variables may improve imputation accuracy across all methods.

References

Christopher M Bishop. Mixture density networks. 1994.

Jesse Bricker, Lisa J Dettling, Alice Henriques, Joanne W Hsu, Lindsay Jacobs, Kevin B Moore, Sarah Pack, John Sabelhaus, Jeffrey Thompson, and Richard A Windle. Changes in us family finances from 2013 to 2016: Evidence from the survey of consumer finances. *Federal Reserve Bulletin*, 103:1–42, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real-nvp. In *International Conference on Learning Representations*, 2017.

Marcello D'Orazio, Marco Di Zio, and Mauro Scanu. *Statistical matching: Theory and practice.* John Wiley & Sons, 2006.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

Mikhail Hushchyn and Egor Zavialov. Probaforms: Conditional generative models for tabular data, 2023. URL `https://github.com/hse-cs/probaforms`. Version 0.2.0. Python library implementing normalizing flows, GANs, and VAEs with scikit-learn interface.

Manu Joseph. Pytorch tabular: A framework for deep learning with tabular data. *arXiv preprint arXiv:2104.13638*, 2021.

María Juaristi and PolicyEngine. Microimpute: Variable imputation methods for survey data, 2025. URL `https://github.com/PolicyEngine/microimpute`. Version 1.8.1. Python library for benchmarking imputation methods including Statistical Matching, OLS, Quantile Regression, and Quantile Random Forests.

Maria Juaristi, Nikhil Woodruff, and Max Ghenis. Methodological advances in microdata imputation: Quantile regression forests for wealth imputation. Working paper, PolicyEngine, 2025. URL `https://github.com/PolicyEngine/microimpute/tree/main/paper`.

Arthur B. Kennickell. Imputation of the 1989 survey of consumer finances: Stochastic relaxation and multiple imputation. In *ASA 1991 Proceedings of the Section on Survey Research Methods*, pages 1–10. American Statistical Association, 1991.

Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica*, pages 33–50, 1978.

Andrey Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4:83–91, 1933.

Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

PolicyEngine. Policyengine us data: Microdata processing for us policy analysis, 2025. URL `https://github.com/PolicyEngine/policyengine-us-data`. Python library for processing US survey microdata including CPS-ASEC, with household-level aggregation, income variable construction, and demographic extraction.

Nickolay Smirnov. Table for estimating the goodness of fit of empirical distributions. *The Annals of Mathematical Statistics*, 19(2):279–281, 1948.

U.S. Census Bureau. Current population survey. `https://www.census.gov/programs-surveys/cps.html`, 2023.

Cédric Villani. Optimal transport: old and new. *Springer*, 2008.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *International Conference on Learning Representations*, 2024.