Benchmarking Deep Learning Methods for Full Variable Imputation

María Juaristi

November 30, 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 generative 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 three deep learning architectures: RealNVP (normalizing flows), Mixture Density Networks (MDN), and TabSyn (VAE with latent diffusion). Using 3-fold cross-validation with quantile loss and distributional accuracy metrics (Wasserstein distance and Kolmogorov-Smirnov statistic), only MDN demonstrates competitive performance. MDN achieves 16% lower median quantile loss than QRF (2.11 ± 0.07 vs. 2.51 ± 0.01), though QRF produces better distributional accuracy with a Wasserstein distance of 472,000 compared to MDN's 1.2 million. RealNVP and TabSyn prove unsuitable in their current configurations, with Wasserstein distances exceeding 12 million, generating distributions that fail to preserve the characteristic shape of wealth data. These findings suggest that while deep generative models show promise for survey imputation, careful architecture selection and hyperparameter tuning are essential, and traditional ensemble methods remain strong baselines for practitioners. Code is available at `https://github.com/juaristi22/cs156-pipeline_2`.

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.

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] systematically benchmarked traditional imputation methods for 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 crucial for policy analysis. Juaristi et al. [2025] call for further comparative studies against deep learning models, 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, effectively learning the underlying data structure through this denoising process. 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 generative models can match or exceed QRF's performance for wealth imputation, and under what conditions each approach excels. In addition to methodological comparison and improvement, success in such an imputation task has direct policy relevance, as accurate wealth imputation enables more precise analysis of wealth taxes, and the distributional consequences of fiscal policy across the income-wealth spectrum. Furthermore, fragmented data sources are common in many fields, which can hinder many different lines of research. Robust imputation methods hence have the potential to enhance the quality of datasets used across various domains where data fragmentation is common, expanding data enhancement practices beyond social science research.

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 policy analysis, where outcomes often 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 generative models for imputation.** We evaluate three deep learning architectures—RealNVP (normalizing flows), Mixture Density Networks, and TabSyn (diffusion-based)—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, 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

This project addresses a fundamental challenge in economic and social science research more broadly. Many important analyses require detailed information at the household level, yet such data are often unavailable in large, representative surveys, instead being 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 highly skewed distribution of wealth. 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, allowing 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 succesfull 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 highly 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 and MDN 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 unkown), 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 three deep generative approaches, each representing a distinct paradigm for conditional density estimation: RealNVP, Mixture Density Networks, and TabSyn. As a baseline, we include Quantile Random Forests (Section 3.5), which prior work established as the best-performing traditional method for this task. 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 changes at every step. This enables exact likelihood computation without needing to approximate the marginal density of the distribution.

Formally, let $z \sim p_Z(z)$ 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 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 a lot of potential in capturing complex, multimodal, and heteroskedastic conditional distributions between predictors and the target variable, in an efficient and 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 tunning this value an important step in ensuring the method is appropirately 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 operalization. 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 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 projection:

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

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.

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:

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

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.

3.4.2 Adaptation for Statistical Matching

The original TabSyn was designed for unconditional data synthesis, with applications in synthetic data generationa 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:

Cross-Dataset Preprocessing. 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.

Masked Diffusion for Conditional Generation. 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.

Imputation Algorithm. The full imputation procedure is summarized in Algorithm 1.

Algorithm 1 TabSyn Cross-Dataset Imputation

Require: Trained VAE encoder E_ϕ, decoder D_θ, diffusion model $\mathcal{D}$
Require: CPS covariates X^{cps}, number of diffusion steps T, trials M
Ensure: Imputed target values $\hat{y}^{\text{cps}}$
Preprocess: Normalize X^{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
    $z^{\text{obs}}_{t-1} \leftarrow z^{\text{obs}}_0 + \sigma_{t-1} \cdot \epsilon$ ▷ Noised observed encoding
    $z_{t-1} \leftarrow (1-m) \odot z^{\text{obs}}_{t-1} + m \odot \tilde{z}_{t-1}$ ▷ 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 $\hat{y}$

3.4.3 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 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 $Q_\tau(Y \mid X)$.

3.5.1 Mathematical Foundation

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

$$\hat{F}(y \mid \mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) \cdot \mathbf{1}_{Y_i \leq y} \tag{26}$$

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:

$$\hat{Q}_\tau(Y \mid \mathbf{x}) = \inf\left\{y : \hat{F}(y \mid \mathbf{x}) \geq \tau\right\} \tag{27}$$

3.5.2 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. Prior work by Juaristi et al. [2025] established QRF as the best-performing traditional method for SCF-to-CPS wealth imputation, making it the baseline against which to evaluate deep learning alternatives.

4 Results and Discussion

4.1 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.

4.1.1 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:

$$\mathcal{L}_\tau(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N} \rho_\tau(y_i - \hat{y}_i) \tag{28}$$

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 five 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.

4.1.2 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:

$$W_1(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(y,\hat{y})\sim\gamma}[|y - \hat{y}|] \tag{29}$$

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:

$$W_1(P, Q) = \int_0^1 |F_P^{-1}(u) - F_Q^{-1}(u)|\, du \tag{30}$$

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 when wanting to use that data to produce reliable policy analysis and distributional impact assessments.

4.1.3 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:

$$D_{n,m} = \sup_x |F_n(x) - G_m(x)| \tag{31}$$

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.

4.2 Cross-Validation Procedure

To evaluate model performance before final imputation, we employ 3-fold cross-validation on the 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 disjoint 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.

4.3 Cross-Validation Results

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

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

Model	$\mathcal{L}_{0.25}$	$\mathcal{L}_{0.50}$	$\mathcal{L}_{0.75}$
QRF (baseline)	2.51 ± 0.03	2.51 ± 0.01	2.46 ± 0.05
MDN	$\mathbf{2.08 \pm 0.07}$	$\mathbf{2.11 \pm 0.07}$	$\mathbf{2.14 \pm 0.07}$
RealNVP	3.17 ± 0.22	3.36 ± 0.17	3.55 ± 0.26
TabSyn	3.49 ± 0.02	6.23 ± 0.06	8.98 ± 0.10

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. The QRF's quantile loss values are moderate, providing a reasonable benchmark against which to evaluate the deep learning approaches.

The Mixture Density Network achieves the best cross-validation performance across all quantile levels, with losses approximately 15–20% lower than the QRF baseline. The MDN's consistent performance across $\mathcal{L}_{0.25}$, $\mathcal{L}_{0.50}$, and $\mathcal{L}_{0.75}$ suggests that the Gaussian mixture parameterization effectively captures the conditional distribution of net worth given the predictor variables. The relatively low standard deviations indicate stable performance across folds.

RealNVP shows moderate 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 tabular regression 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.

4.4 Final Imputation Results

After cross-validation, each model was trained on the full SCF dataset and used to impute net worth onto the CPS. Table 3 presents two distributional accuracy metrics comparing the imputed CPS distributions to the SCF donor distribution.

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

Model	Wasserstein Distance	KS Statistic
QRF (baseline)	**471,900**	**0.041**
MDN	1,199,237	0.049
RealNVP	13,809,212,015	0.230
TabSyn	12,665,210	0.168

The distributional accuracy results shed important light on each model's ability to preserve the overall wealth distribution in the imputed CPS data, complementing the cross-validation measure of conditional accuracy. Despite achieving the best CV quantile loss, MDN produces an imputed distribution with Wasserstein distance 2.5 times larger than QRF. Nonetheless, it is still the model with best distributional accuracy out of the deep learning methodologies studied. More dramatically, RealNVP and TabSyn produce a Wasserstein distance four orders of magnitude larger than QRF, indicating very poor distributional preservation. The KS statistic provides a complementary perspective. QRF achieves the lowest KS statistic (0.041), indicating that its imputed CDF closely tracks the SCF distribution. MDN follows closely (0.049), while TabSyn (0.168) and RealNVP (0.230) show substantially larger maximum CDF deviations.

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-simetrical 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.

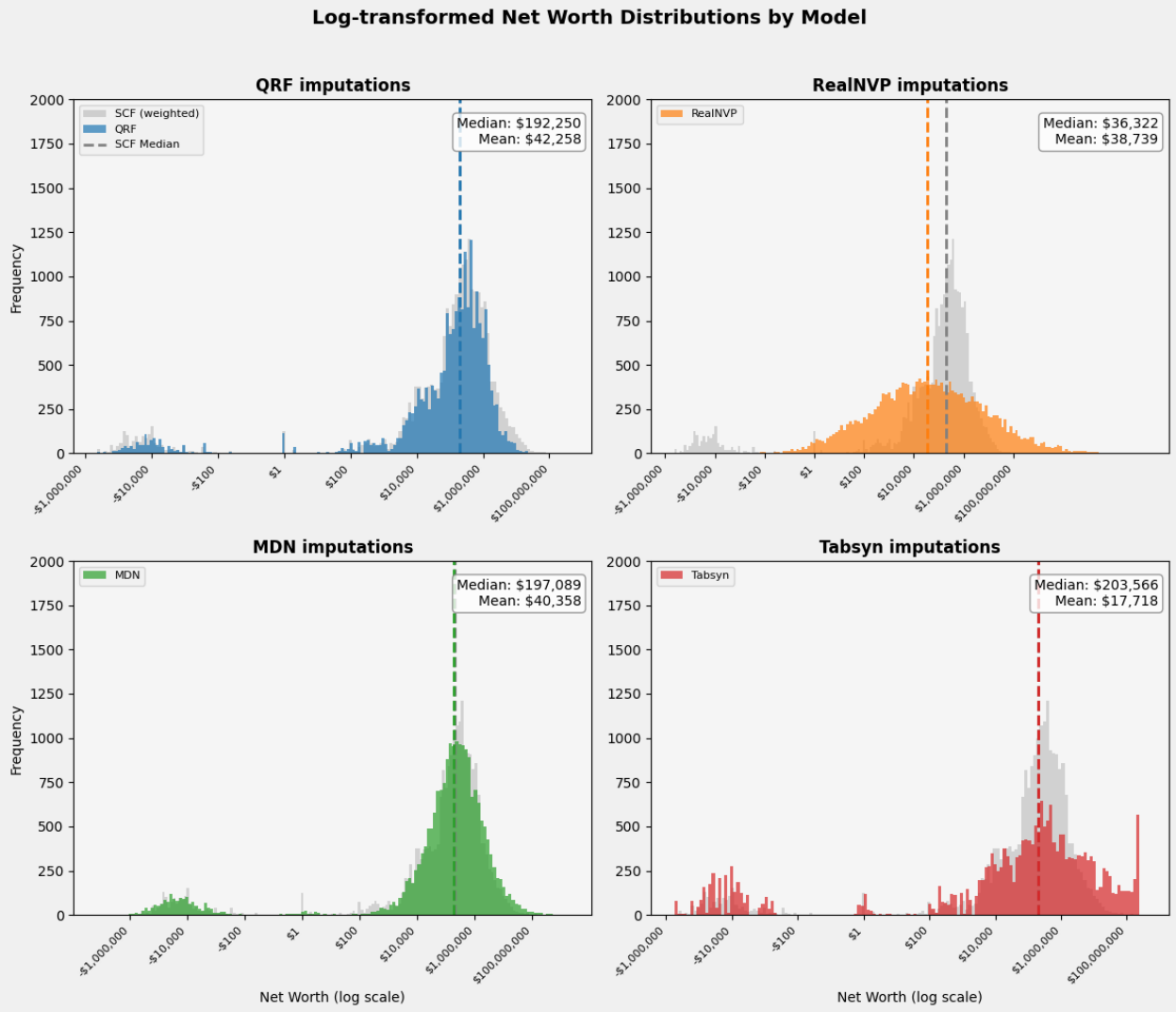

Figure 1: Log-transformed net worth weighted distributions comparing SCF donor data (grey) against imputed CPS values for each model. Dashed lines indicate median values. QRF and MDN closely match the SCF distribution shape and median, while RealNVP exhibits a shifted, symmetric distribution missing the characteristic right skew. TabSyn captures the general shape but over-generates extreme values in both tails.

4.5 Discussion

Among the deep learning approaches evaluated, only the Mixture Density Network demonstrates performance comparable to the QRF baseline. MDN achieves the best cross-validation quantile loss and produces distributional accuracy metrics (Wasserstein distance and KS statistic) within the same order of magnitude as QRF. While MDN does not significantly outperform QRF on distributional metrics, its competitive performance suggests that neural network-based density estimation can be viable for survey imputation, potentially requiring additional work for hyperparameter tuning and more careful configuration.

In contrast, RealNVP and TabSyn seem to be unsuitable for this task with their current configurations, and working toward improving their architectures and training procedures may be

unnecessarily complex for marginal improvement. RealNVP produces a nearly symmetric distribution centered far below the true median, completely failing to capture the characteristic right skew of the wealth distribution. TabSyn, despite its sophisticated VAE-diffusion architecture, generates implausible extreme values.

The relative success of MDN compared to the other deep learning methods may stem from its explicit probabilistic formulation. The Gaussian mixture parameterization directly models the conditional density $p(y|x)$, providing interpretable components that can capture multimodality and heteroskedasticity. Normalizing flows (RealNVP) and diffusion models (TabSyn), while theoretically more flexible, may require more careful tuning or architectural modifications to handle the specific challenges of wealth data.

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. Given the competitive performance of MDN, its moderate training time may be justifiable for researchers seeking a deep learning alternative when seeking flexibility and interpretability.

TabSyn is by far the most 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 Implications for Survey Data Fusion

Our findings suggest that while most deep generative models tested here fail to outperform traditional methods, MDN represents a viable alternative worthy of further investigation. The disconnect between CV loss and distributional accuracy observed across models highlights that standard model selection criteria may be insufficient for imputation tasks where preserving marginal distributions is essential.

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.

- **Hyperparameter sensitivity**: The deep learning models' performance depends on architecture and training choices that were not exhaustively optimized in this study.

Given MDN's promising results, future work should explore whether alternative hyperparameter configurations could enable it to surpass QRF. Potential directions include:

- Increasing the number of Gaussian mixture components beyond the five used here, potentially allowing better capture of the wealth distribution's heavy tails.
- Experimenting with alternative mixture distributions (e.g., Student-t components) that naturally accommodate heavier tails than Gaussians.
- Incorporating regularization techniques specifically designed to prevent the distribution compression observed in our results.
- Exploring deeper or wider network architectures to improve the network's capacity to learn complex conditional relationships.
- Evaluating whether increased training time allows for significant performance improvements.

5 Conclusion

5.1 Summary

This paper investigated whether deep generative 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 three deep learning architectures—RealNVP (normalizing flows), Mixture Density Networks, and TabSyn (VAE with latent diffusion)—against a QRF baseline, using both cross-validation quantile loss and distributional accuracy metrics.

Despite being a traditional ensemble method, QRF remains the strongest overall performer, achieving the best distributional accuracy (Wasserstein distance of 472,000; KS statistic of 0.041) while maintaining competitive cross-validation performance. Among the deep learning approaches, only MDN demonstrates competitive performance. MDN achieves 16% lower median quantile loss than QRF and produces distributional metrics within the same order of magnitude. Its explicit Gaussian mixture parameterization provides interpretable density estimation that generalizes reasonably well to the receiver dataset, suggesting that neural network-based methods can be viable for survey imputation with more fine-tuned architectural choices.

In contrast, RealNVP and TabSyn prove unsuitable, at least with their current configurations. Both models produce imputed distributions with Wasserstein distances exceeding 12 million, which are orders of magnitude worse than QRF. RealNVP generates a symmetric distribution missing the characteristic right skew of wealth, while TabSyn over-generates extreme values in both tails. These failures highlight the challenges of applying flexible generative models to heavy-tailed survey data without careful adaptation. The heavy computational burdern of TabSyn further limits its practicality for this task, while the reasonable training time of MDN makes it a more accessible deep learning alternative for researchers seeking a very flexible imputation approach.

Overall, MDN may become a promising model as it seems to competitively capture complex distributions like wealth, while offering additional flexibility, which may be beneficial in other imputation contexts. Nonetheless, they remain a method with higher training data and computational demands compared to QRF, which continues to serve as a robust, efficient baseline for survey imputation tasks where preserving distributional characteristics.

5.2 Pipeline Overview

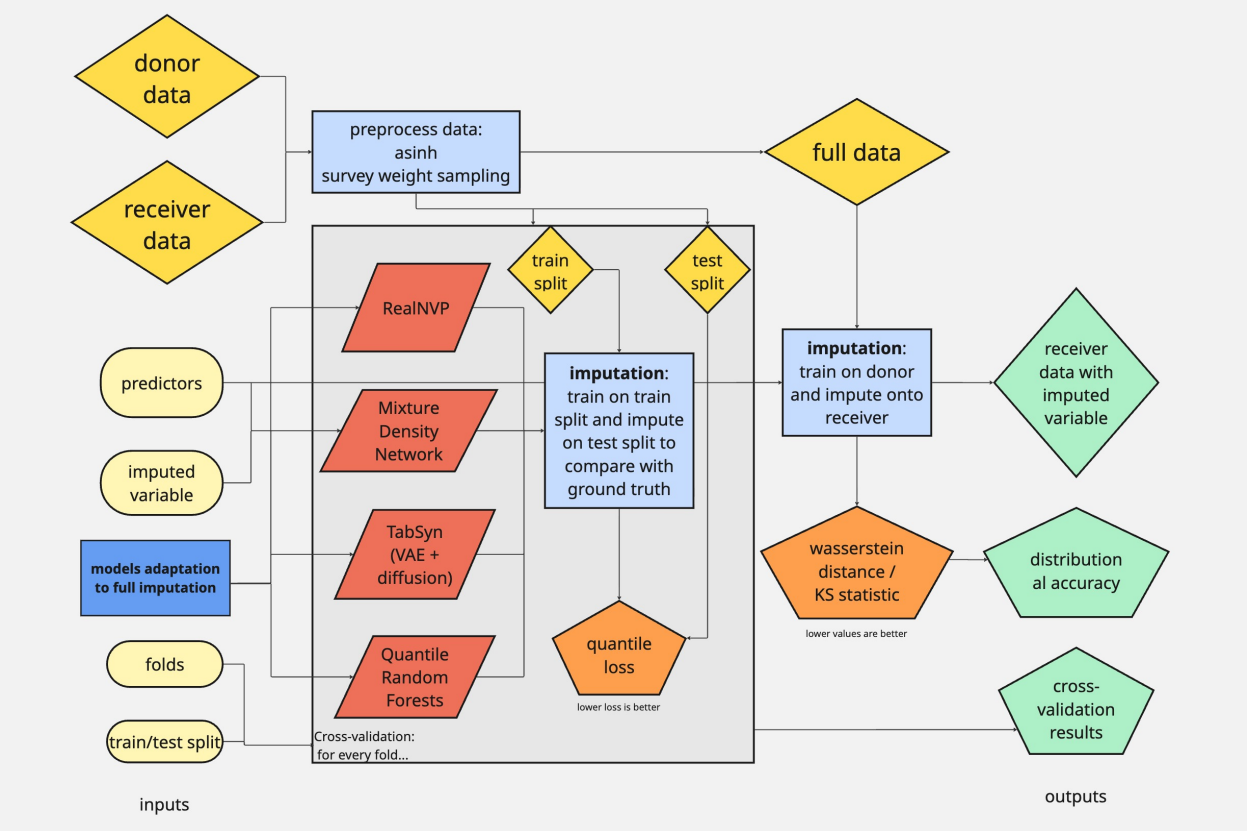

Figure 2: Overview of the imputation and model benchmarking pipeline.

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 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_2`, 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 MDN can surpass QRF with alternative configurations, including more mixture components, heavier-tailed distributions (e.g., Student-t mixtures), and deeper network architectures. Additionally, investigating why normalizing flows and diffusion models struggle with this task could inform architectural modifications that better handle heavy-tailed survey data. 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.

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`.

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.