Personal Total Income of Individual in the United States of America in 2023*

A Predictive Model Using the American Community Survey Data

Justin (Jiazhou) Bi and Weiyang Li

October 25, 2024

This project aims to build a predictive model for personal total income using various demographic attributes, such as marital status, occupation, and residing state. We explored three popular machine learning algorithms: Random Forest, Linear Regression, and Extreme Gradient Boosting. Ultimately, Extreme Gradient Boosting was selected due to its superior predictive performance. While the model achieved a high R-squared value, it also exhibited a high Mean Squared Error, indicating challenges in accurately predicting all data points. Future steps for improving the model may involve incorporating additional demographic features to enhance model accuracy and reduce errors. Additionally, fine-tuning hyperparameters, experimenting with feature engineering, and handling outliers could further improve performance.

1 Introduction

Income is a fundamental concept as it affects nearly every member of society, influencing access to resources, quality of life, and overall economic stability. At the same time, it is also a highly complex concept to define, as it can be interpreted differently depending on various perspectives, such as economic, social, legal, and tax purposes. Therefore, we must examine various demographical factors that may impact income to understand its concept better. Our project defines income as "all forms of net financial resources generated or lost from work, investments, or other activities." This project aims to predict personal income using data from the 2023 American Community Survey (ACS) (Ruggles et al. (2024)). The ACS provides detailed demographic information, including age, education, employment, marital

*Inspired and instructed by: https://github.com/RohanAlexander/marriage. All the project related files can be found at: https://github.com/Jiazhou-Bi/ACS-Income-Model/tree/main.

status, mortgage status, veteran status, etc… As a result, it provides a robust dataset for studying potential factors that may influence income.

Due to the timeframe limitations, the findings of this project are specifically applicable to the 2023 dataset. As such, the results may vary across different periods, and their generalizability has not been assessed within this project. The raw data used in this analysis was sourced from IPUMS using the IPUMS API. (IPUMS (2024)) We utilized pandas (McKinney (2010)) for data processing, and data visualization was carried out using Seaborn. (Waskom (2021)) Additionally, all tables presented in this report were generated with Plotly. (Inc. (2015))

Income inequality is a serious problem and can lead to significant consequences and it may affect social structure, economic stability, as well as policy-making processes. Researchers found that demographic factors, such as education level and occupation type, can significantly impact an individual's financial standing. (Autor (2013)) Similarly, social relationships, such as marital status and household characteristics, affect income dynamics. (Waite and Gallagher (2000)) Therefore, applying machine learning models to analyze the ACS dataset and using the information mentioned to predict personal income is reasonable.

We have chosen 2023 ACS dataseet because it offers many demographical details, as well as self-reported total income of the individuals. The sample is also representative and covers all geographical locations in the USA. Furthermore, it provides easy access to their database through API. We will employ advanced predictive models, including random forest, lineaer regression, and extreme gradient boosting machine to estimate personal income based on demographic factors to provide insights into which potential factors have the most significant impact on personal income. We hope the findings generated from our project can provide insights to policy-making process so that we can, hopefully, reduce income inequality and support further economic development in the future.

If we can successfully predict personal income using demographic information from the survey, this analysis will be able to contribute to a nuanced understanding of the American economy in 2023. It can help to reveal how demographic changess, educational attainment, and employment status/type can impact personal income. On the other hand, if the model cannot predict personal income with a high degree of confidence, this would suggest that the factors included in the analysiscannot not fully capture the intricacies of economic behavior and individual circumstances. As a result, additional variables, such as detailed education background like school of graduation, languages spoken, health status, etc, might be necessary in accurately predicting income.

2 Data

The dataset used here is downloaded from the Toronto Open Data website via (**TorontoOpenData?**). This dataset contains all the reported crimes that happened in Toronto from 2014 to 2023. This dataset is grouped by the year of the reported crime, its category and belonging subtype,

and the count of the subtype being reported and cleared for that year for each division. Because I am examining the crime pattern in the city, I have dropped the division information and aggregated the existing data according to their subtype and the year of the crime being reported. In the following subsections, I will review all the variables used in this report and provide some basic descriptive statistics. The first five rows of the cleaned data used for analysis are attached (Table 1).

Table 1: Example of Cleaned Data

	STATEICP	GQ	OWNERSHP	MORTGAGE	SEX	AGE	MARST	EDUC	SCHLTYPE	OCC
1	41	1	1	3	2	51	6	10	1	800
2	41	1	1	3	1	61	1	8	1	4810
3	41	1	1	1	1	63	1	7	1	8030
4	41	1	1	1	2	36	4	7	1	120
5	41	1	1	1	1	17	6	5	3	4000

2.1 Report Year

The report year variable is the number of crimes being reported. In this dataset, the data spans from 2014 to 2023, encompassing ten years. No month or date information was given; thus, there are only ten different values for this variable in chronological order.

2.2 Category

The category includes information about the nature of the crime. There are six crime categories: Crimes Against Property, Crimes Against the Person, Other Federal Statute Violations, Other Criminal Code Violations, Controlled Drugs and Substances Act, and Criminal Code Traffic. They are listed in the table below (**?@tbl-table2**).

2.3 Subtype

There exist multiple subtypes under each crime category. The following is an exhaustive table (**?@tbl-table3**) of all crimes' subtypes and their respective category.

2.4 Count

In the original table, this value is grouped by the subtype of the crime, the division, and the year when the crime was reported. The original count indicates the number of a specific subtype of crime reported within a particular division for the year. However, as mentioned

before, because I am only interested in all the crimes in the City of Toronto, I have dropped the division information and aggregated the count from all the divisions to a single value. Therefore, for each subtype of the crimes, a total count of that subtype is reported in a single year.

2.5 Count_Cleared

These are the counts of crimes identified as cleared. In plain words, these are crimes that are dealt/solved. I have taken the same approach for this column as the previous one. After cleaning the data,f, or each subtype of the crimes, there is a total count of that subtype being reported that is also cleared in a single year.

2.6 Case_Clearing_Rate

This column was not included in the raw dataset but was created by dividing the cleared crimes by total crimes. A higher case-clearing rate for a particular subtype of crime usually suggests a higher effectiveness of law enforcement in dealing with this subtype of crime. The value is ranged from 0 to 100%.

3 Exploratory Data Analysis

4 Model

Our modeling strategy aims to explore the relationship between personal characteristics and income. We implemented three models: Linear Regression with Interaction Terms, Random Forest, and XGBoost, each chosen for their unique strengths in prediction and interpretation.

Background details are included in Appendix Section 7.2.

4.1 Linear Regression Model with Interaction Term

4.1.1 Model set-up

We implemented a linear regression model with an interaction term to explore the relationship between personal characteristics and income. The general form of the model is:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \beta_{\text{interaction}}(X_i \times X_j) + \epsilon$$

where:

- y represents the total income (`INCTOT`),
- $X_1, X_2, ..., X_n$ are the predictor variables, including both categorical and numerical features,
- $X_i \times X_j$ is the interaction term between education and sex (`EDUC_SEX_INTERACTION`)
- β_0 is the intercept,
- $\beta_1, \beta_2, ..., \beta_n, \beta_{\text{interaction}}$ are the coefficients for the respective predictors and interaction term.

To ensure the robustness of our linear regression model, we conducted a series of steps aimed at minimizing multicollinearity and improving model interpretability. Initially, we included all relevant predictor variables, both categorical and numerical, to explore the full range of relationships between personal characteristics and income. Next, we calculated the Variance Inflation Factor (VIF) for each predictor to identify potential multicollinearity issues. Typically, a VIF value greater than 10 indicates problematic multicollinearity, which can distort the model's estimates. In our analysis, variables such as `VESTAT` (Veteran Status) and `AGE` had VIF values exceeding 13, suggesting they were highly correlated with other predictors. To address this, we removed these high-VIF variables from the model. Afterward, we refit the linear regression model using the reduced set of predictors, which resulted in improved performance metrics, including a lower Mean Squared Error and a more interpretable set of coefficients.

Note that according to our previous analysis in Section 3, the interaction term, `EDUC_SEX_INTERACTION`, was included to capture the combined effect of education level and gender on income. This allowed us to model the non-linear relationship between these two variables and their joint impact on the outcome variable.

Table 1: Coefficients from the Linear Regression Model

Feature	Coefficient
`cat__MORTGAGE`	11267.90
`num__AGE`	415.91
`cat__STATEICP`	126.39
`cat__IND1990`	-43.18
`cat__OCC2010`	-139.55
`num__EDUC_SEX_INTERACTION`	-1832.32
`cat__MARST`	-5541.39
`cat__GQ`	-7956.89
`cat__SCHLTYPE`	-16320.74

Table 1: Linear Regression Coefficients Table.

We run the model in Python (Python Software Foundation 2024) using the `scikit-learn` package (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024). We use the default settings for the `LinearRegression` function from `scikit-learn`. We use the `statsmodels` package (Seabold, Skipper and Perktold, Josef 2024) to calculate the VIF for each predictor.

4.1.2 Model Justification

In evaluating our model's performance, we selected key metrics to assess both its fit and predictive accuracy, including the Variance Inflation Factor (VIF), R-squared (R^2), and Mean Squared Error (MSE). These metrics were chosen to provide a comprehensive evaluation of the model's robustness and reliability in capturing the relationship between personal characteristics and income.

The VIF was used to diagnose multicollinearity among predictors, helping us ensure that no variables were highly correlated, which could distort the interpretation of the model's coefficients.

Table 2: Variance Inflation Factors (VIF) for Predictor Variables

Feature	VIF
cat___STATEICP	3.242254
cat___GQ	1.002419
cat___MORTGAGE	2.658013
cat___MARST	1.741158
cat___SCHLTYPE	1.239412
cat___OCC2010	3.187731
cat___IND1990	8.490997
num___AGE	7.871046
num___EDUC_SEX_INTERACTION	6.945850

Table 2: This table displays the Variance Inflation Factors (VIF) for each of the predictor variables in the model. VIF values greater than 10 indicate potential multicollinearity issues.

We chose R^2 as it provides a measure of how well the model explains the variance in the dependent variable, making it an important indicator of the model's explanatory power. The

MSE was selected to quantify the average squared difference between observed and predicted values, offering a clear view of the model's prediction error.

Table 3: Model Performance Metrics

Metric	Value
R-squared	0.0774938018
Adjusted R-squared	0.0774637241
Mean Squared Error (MSE)	7,448,800,507.01
Root Mean Squared Error (RMSE)	86,306.43

Table 3: This table presents the performance metrics of the linear regression model, including MSE, RMSE, R-squared, and Adjusted R-squared. These metrics are used to assess the model's fit and predictive accuracy.

Additionally, residual analysis was conducted by plotting residuals against fitted values to check for randomness around zero, ensuring that the model satisfied the assumptions of linearity and homoscedasticity.

We use the statsmodels package (Seabold, Skipper and Perktold, Josef 2024) to calculate the VIF for each predictor. We use the matplotlib (Hunter and Droettboom 2024) and seaborn (Waskom 2021) to visualize the residuals. We use the scikit-learn (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024) package to calculate the R^2 score.

4.2 Random Forest Model

4.2.1 Model set-up

To analyze the non-linear relationship between personal characteristics and income, we employed a Random Forest model. Random Forest is an ensemble learning method that builds multiple decision trees and averages their predictions, reducing overfitting and capturing complex, non-linear relationships in the data. We utilized Bagging (Bootstrap Aggregating) and random feature selection to enhance model performance and reduce overfitting. Bagging involves generating multiple bootstrap samples from the original training set, with each sample used to train an independent decision tree. By averaging the predictions of multiple trees, Bagging reduces the variance of the model, thereby improving its robustness. Additionally, Random Forest introduces further randomness by selecting a subset of features at each split, which reduces correlation among trees and improves the model's generalization ability.

To optimize the model, we conducted hyperparameter tuning, focusing on two key parameters: `n_estimators` (the number of trees) and `max_depth` (the maximum depth of each tree). Increasing `n_estimators` typically decreases model variance and improves stability, but too many trees can result in diminishing returns with increased computation time. Initially, we set `n_estimators` to 100, but through Grid Search, we tested values of 100, 200, and 300 and found that 300 trees yielded the best performance. Similarly, the `max_depth` parameter controls the complexity of the trees. While deeper trees can capture more intricate data patterns, they may also overfit. We initially set `max_depth` to 15 and test 10, 15, 20 but found that reducing it to 10 during hyperparameter tuning improved generalization by avoiding overfitting. Additionally, we tuned `min_samples_split` and `min_samples_leaf` to prevent overfitting by ensuring that nodes have a sufficient number of samples before splitting.

We run the model in Python (Python Software Foundation 2024) using the scikit-learn package (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024) and pandas (McKinney 2010). We used the GridSearchCV function from scikit-learn for hyperparameter tuning and subsampled the training data using pandas. The RandomForestRegressor function from scikit-learn was employed for model fitting.

4.2.2 Model Justification

To assess the performance of our Random Forest model, we evaluated several key metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), feature importance, and residual analysis. These evaluations provided insights into the model's predictive accuracy, interpretability, and overall fit.

First, we calculated the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to assess the overall prediction accuracy of the model. These metrics provided insight into how well the model captured the variance in income based on the selected features.

Table 4: Best Random Forest Model Performance Metrics

Metric	Value
Best Model MSE	6,057,429,750.02
Best Model RMSE	77,829.49

Table 4: This table shows the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) for the best-tuned Random Forest model, demonstrating the model's performance after hyperparameter tuning.

Additionally, we performed a feature importance analysis, which allowed us to understand the relative significance of each predictor variable in determining income. This is particularly useful for verifying that the model identifies the key factors influencing income as expected.

Table 5: Feature Importance in the Best Random Forest Model

Feature	Importance
cat__EDUC_new	0.337065
cat__IND1990	0.243454
num__AGE	0.193926
cat__SEX	0.149342
cat__MARST	0.055948
cat__MORTGAGE	0.013064
cat__SCHLTYPE	0.004251
cat__VETSTAT	0.002948
cat__OWNERSHP	0.000000

Table 5: This table lists the feature importance values for the best-performing Random Forest model. The higher the importance value, the more significant the feature is in predicting the target variable (income).

We also conducted a residual analysis by plotting the residuals (the difference between actual and predicted values) against the predicted values. This step helped us evaluate whether the model appropriately captured the underlying patterns in the data, ensuring that there were no significant biases or violations of model assumptions.

The feature importances were extracted from the best model using RandomForestRegressor from scikit-learn (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024) and displayed using pandas (McKinney 2010). We use the matplotlib (Hunter and Droettboom 2024) to visualize the residuals.

4.3 XGBoost

4.3.1 Model set-up

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm that builds an ensemble of decision trees using gradient boosting techniques. In gradient boosting, each subsequent tree is trained to correct the residuals of the previous trees, iteratively improving the

model's predictive accuracy. XGBoost enhances this process by incorporating regularization techniques to control overfitting, making it an ideal choice for capturing complex, non-linear relationships in data.

To optimize the performance of our XGBoost model, we employed GridSearchCV to conduct hyperparameter tuning. We explored a range of hyperparameters to find the optimal configuration for our dataset. The grid search included the learning rate (`learning_rata`), max depth (`max_depth`), number of trees (`n_estimators`), and minimum child weight (`min_child_weight`). Specifically, we tested learning rates of 0.01, 0.05, and 0.1, tree depths of 4, 6, and 8, and `n_estimators` values of 100, 200, and 300. Using GridSearchCV with a cross-validation fold of 3, we systematically evaluated different combinations of these hyperparameters. The model was evaluated using negative Mean Squared Error (MSE) as the scoring metric. The optimal model used a learning rate of `0.1`, a maximum tree depth of `6`, a minimum child weight of `5`, and `300` boosting rounds. This configuration provided a good balance between model complexity and predictive accuracy, capturing important non-linear relationships while minimizing overfitting.

We implemented the XGBoost model in Python (Python Software Foundation 2024) using the scikit-learn package (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024), with pandas (McKinney 2010) for data processing and xgboost (Cho and Yuan 2024) for model fitting. The initial model was trained using default parameters of the XGBRegressor function, which was subsequently fine-tuned using GridSearchCV from scikit-learn (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024) for hyperparameter optimization.

4.3.2 Model Justification

Similar to the Random Forest model, the XGBoost model was trained on a 70-30 train-test split. We evaluated the model's performance using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to assess its predictive accuracy.

Table 6: XGBoost Model Performance Metrics

Metric	Value
Mean Squared Error (MSE)	5,191,183,577.52
Root Mean Squared Error (RMSE)	72,049.87

Table 6: This table shows the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) for the final XGBoost model, reflecting its predictive accuracy.

We performed a residual analysis by plotting the residuals against the predicted values to assess the model's fit. This allowed us to check whether the XGBoost model accurately captured the patterns in the data and to verify that no major biases or violations of model assumptions were present.

We also performed cross-validation to ensure the robustness of the XGBoost model. This technique allowed us to evaluate the model's performance across different subsets of the data, helping to confirm that the model generalizes well and is not overly sensitive to any particular portion of the training data.

Table 7: XGBoost Cross-Validation Performance

Metric	Value
Cross-Validation RMSE	71,994.55

Table 7: This table shows the Root Mean Squared Error (RMSE) from cross-validation, demonstrating the model's performance across different subsets of the data.

We used matplotlib (Hunter and Droettboom 2024) to perform a residual analysis against the predicted values from the XGBoost model. We performed cross-validation using cross_val_score function from scikit-learn (Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024) to evaluate the robustness of the model.

5 Results

5.1 Liner Regression Model

6 Discussion

6.1 Weaknesses and next steps

7 Appendix

7.1 Data Cleaning

7.2 Model Details

References

Autor, David H. 2013. "The "Task Approach" to Labor Markets: An Overview." *Journal for Labor Market Research* 46 (3): 185–99. https://doi.org/10.1007/s12651-013-0125-7.

Cho, Hyunsu, and Jiaming Yuan. 2024. *XGBoost Python Package*. XGBoost Developers. https://xgboost.ai/.

Hunter, John D., and Michael Droettboom. 2024. *Matplotlib: Python Plotting Package*. Python Software Foundation. https://matplotlib.org.

Inc., Plotly Technologies. 2015. "Collaborative Data Science." Montreal, QC: Plotly Technologies Inc. 2015. https://plot.ly.

IPUMS. 2024. "IPUMS API." IPUMS; Data retrieved via IPUMS API.

McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, edited by Stéfan van der Walt and Jarrod Millman, 56–61. https://doi.org/ 10.25080/Majora-92bf1922-00a .

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2024. *Scikit-Learn: Machine Learning in Python*. Python Software Foundation. http://scikit-learn.org.

Python Software Foundation. 2024. *Python: A Dynamic, Open Source Programming Language*. Python Software Foundation. https://www.python.org/.

Ruggles, Steven, Sarah Flood, Matthew Sobek, Daniel Backman, Annie Chen, Grace Cooper, Stephanie Richards, Renae Rodgers, and Megan Schouweiler. 2024. "IPUMS USA: Version 15.0 [dataset]." Minneapolis, MN: IPUMS. https://doi.org/10.18128/D010.V15.0.

Seabold, Skipper and Perktold, Josef. 2024. *Statsmodels: Statistical Computations and Models for Python*. Python Software Foundation. https://www.statsmodels.org/.

Waite, Linda J., and Maggie Gallagher. 2000. *The Case for Marriage: Why Married People Are Happier, Healthier, and Better Off Financially*. New York, NY: Broadway Books.

Waskom, Michael L. 2021. "Seaborn: Statistical Data Visualization." *Journal of Open Source Software* 6 (60): 3021. https://doi.org/10.21105/joss.03021.