BZAN 542 - Fall 2023

Predictive Modeling for Income Estimation

Group:

Ian Sebby
Hannah Johnson
Aaron Harper

Goal

This report aims to leverage the census and survey data provided by the Integrated Public Use Microdata Series (IPUMS) to employ statistical analysis and machine learning techniques to predict an individual's income status. This approach is particularly useful for understanding the effects on income across different demographic groups and geographical regions.

The practical application of this analysis can serve many industries including, but not limited to:

Financial Services

- Risk management:
    - Incorporate income predictions to aid in risk mitigation and more informed decision-making for the financial sector.
- Financial inclusion:
    - Develop strategies for expanding financial services to underserved populations based on income.

Marketing

- Customer Relationship Management (CRM):
    - Tailor marketing efforts based on predicted income levels to offer a more personalized business-to-consumer interaction.
- Forecast sales:
    - Improve sales by integrating income predictions into forecasting models.
- Diversification of products:
    - Identifying market segments and offering product diversification based on trends in predicted income.

Academic Research

- Identify trends or anomalies:
    - Find anomalies related to income variations and provide insights for research.
- Target groups of study
    - Recognize trends in demographic groups for more in-depth research related to disparities in income.

Data Overview

The dataset used in this analysis was a sample retrieved from IPUMS, a leading organization that provides access to global social and economic data. The data is from the American Community Survey (ACS), which conducts yearly surveys similar to the Census. The sample included a total 205,912 individuals (rows) and 13 self-reported variables.

The reported attributes can be seen below.

Attribute	Description
REGION	Identifies the region and division where the housing unit was located.
STATEFIP	Reports the state in which the household was located, using the Federal Information Processing Standards (FIPS) coding scheme, which orders the states alphabetically.
METRO	Indicates whether the household resided within a metropolitan area and, for households in metropolitan areas, whether the household resided within or outside of a central/principal city.
FAMSIZE	Counts the number of own family members residing with each individual, including the person her/himself.
SEX	Reports whether the person was male or female.
AGE	Reports the person's age in years as of the last birthday.
MARST	Gives each person's current marital status.
RACE	Provides the full detail given by the respondent and/or released by the Census Bureau.
CITIZEN	Reports the citizenship status of respondents, distinguishing between naturalized citizens and non-citizens.
EDUCD	Indicates respondents' educational attainment, as measured by the highest year of school or degree completed.
EMPSTAT	Indicates whether the respondent was a part of the labor force -- working or seeking work -- and, if so, whether the person was currently unemployed.
UHRSWORK	Reports the number of hours per week that the respondent usually worked, if the person worked during the previous year.
INCWAGE	Reports each respondent's total pre-tax wage and salary income - that is, money received as an employee - for the previous year.

Data Preprocessing

Filtering

- Filtered data to those who have AGE greater than 18 and EMPSTAT = 1, implying the individual is employed.

Relabeling

- All variables were converted from the IPUMS listed codes to appropriate categorical labels.

Missing Values

- Removed rows with missing values in the METRO variable.

Ordered Factors

- Made EDUCD an ordered factor to help the models make more informed and accurate predictions.

Near Zero Variance

- All variables were checked for near zero variance to reduce redundancy and improve model performance. Only one variable, EMPSTAT, had near zero variance which was due to the data being filtered to employed individuals only. This variable was removed from the dataset before training the models.

Discretizing

- INCWAGE was changed from a continuous variable to bins with 2 levels (Above Median and Below Median) that indicate if the individual was above or below the median income.

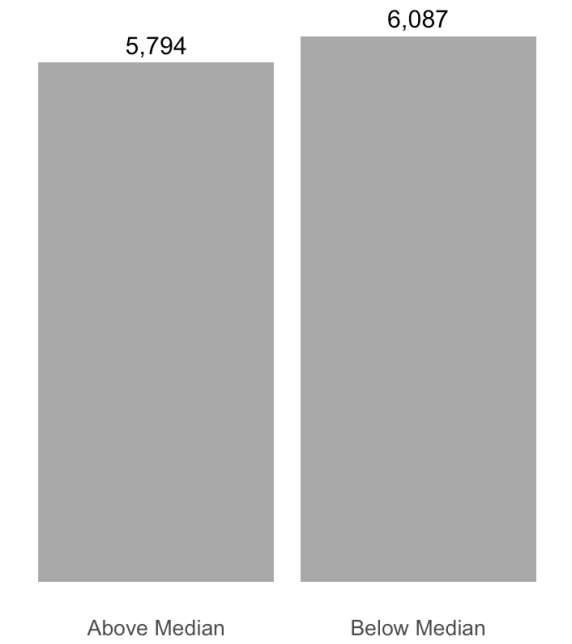
Sampling

- Additionally, due to its size, a smaller subset of the dataset was taken via a random sample to remove the computational burden during predictive modeling. The original size of the dataset included 205,912 observations but only about 11,881 observations were used to train and test the predictive models.

Model Development

A series of models were made to predict whether an individual would be above or below the median income ($45,000) within the dataset. Initially, about 48.8% of individuals are above the median income and 51.2% are below the median income.

Figure 1



Therefore a naive model that uses majority class predictions might predict all individuals to be below the median income, which would be only 51.2% accurate. However, by developing a predictive model, we were able to predict whether an individual is above or below the median income with about a 77% accuracy.

The models used for predictions were an XGBoost, GBM, Random Forest, and a SVM model.

Below are tuning parameters for each model sorted based on their accuracies. There is no preference for the majority of these tunes since most accuracies fall within one standard deviation of each other. Therefore the selected tune was chosen by the highest accuracy for every model in combination with the one standard deviation rule, but other tuning parameters would provide similar results if chosen instead.

GBM Tuning

```
> print(GBM_accuracies[, selected_columns])
# A tibble: 54 × 9
# Groups:   shrinkage, interaction.depth, n.minobsinnode, n.trees, Accuracy, AccuracySD [54]
   shrinkage interaction.depth n.minobsinnode n.trees Accuracy AccuracySD Lower_Bound Upper_Bound SD_Difference
       <dbl>             <dbl>          <dbl>   <dbl>    <dbl>      <dbl>       <dbl>       <dbl>         <dbl>
 1      0.01                 3             10    1000    0.767    0.00930       0.758       0.776        0.0186
 2      0.01                 7             10     750    0.767    0.0115        0.755       0.778        0.0229
 3      0.01                 7             10    1000    0.767    0.0125        0.754       0.779        0.0249
 4      0.1                  5             10     500    0.767    0.0163        0.750       0.783        0.0327
 5      0.01                 5              5    1000    0.766    0.00967       0.756       0.776        0.0193
 6      0.01                 5             10    1000    0.766    0.0121        0.754       0.778        0.0241
 7      0.1                  3             10     500    0.765    0.00970       0.756       0.775        0.0194
 8      0.01                 5             10     750    0.765    0.0111        0.754       0.776        0.0222
 9      0.01                 7              5    1000    0.765    0.0111        0.754       0.776        0.0222
10      0.01                 3              5    1000    0.765    0.00968       0.755       0.775        0.0194
```

Random Forest Tuning

```
> print(FOREST_accuracies[, selected_columns])
# A tibble: 4 × 6
# Groups:   mtry, Accuracy, AccuracySD [4]
   mtry Accuracy AccuracySD Lower_Bound Upper_Bound SD_Difference
  <dbl>    <dbl>      <dbl>       <dbl>       <dbl>         <dbl>
1    12    0.764    0.00412       0.760       0.768       0.00825
2     5    0.764    0.00792       0.756       0.772       0.0158
3     3    0.749    0.00693       0.742       0.756       0.0139
4     1    0.702    0.00721       0.695       0.709       0.0144
```

XGBoost Tuning

```
> print(XGBOOST_accuracies[, selected_columns])
# A tibble: 135 × 12
# Groups:   eta, max_depth, gamma, colsample_bytree, min_child_weight, subsample, nrounds, Accuracy, AccuracySD [135]
     eta max_depth gamma colsample_bytree min_child_weight subsample nrounds Accuracy AccuracySD Lower_Bound Upper_Bound SD_Difference
   <dbl>     <dbl> <dbl>            <dbl>            <dbl>     <dbl>   <dbl>    <dbl>      <dbl>       <dbl>       <dbl>         <dbl>
 1   0.1         6   5                0.6                1       0.8     500    0.770    0.00697       0.763       0.777        0.0139
 2   0.1         3   1                0.6                1       0.8     500    0.769    0.00648       0.762       0.775        0.0130
 3   0.1         3   5                0.8                1       0.8     500    0.768    0.00942       0.759       0.778        0.0188
 4   0.1         3   0.1              0.6                1       0.6     500    0.768    0.00943       0.759       0.778        0.0189
 5   0.1         3   5                1                  1       0.6     500    0.768    0.00906       0.759       0.777        0.0181
 6   0.1         6  10                0.8                1       0.6     500    0.768    0.0103        0.758       0.778        0.0206
 7   0.1         3   0.1              0.6                1       0.8     500    0.768    0.0107        0.757       0.779        0.0213
 8   0.1         3   1                1                  1       0.8     500    0.768    0.00476       0.763       0.773        0.00951
 9   0.1         3   0.5              1                  1       0.6     500    0.768    0.0103        0.758       0.778        0.0207
10   0.1         3   5                0.8                1       1       500    0.768    0.0116        0.756       0.779        0.0233
```

SVM Tuning

```
> print(SVM_accuracies[, selected_columns])
# A tibble: 7 × 6
# Groups:   C, Accuracy, AccuracySD [7]
      C Accuracy AccuracySD Lower_Bound Upper_Bound SD_Difference
  <dbl>    <dbl>      <dbl>       <dbl>       <dbl>         <dbl>
1   256    0.749     0.0150       0.734       0.764        0.0299
2    64    0.748     0.0157       0.732       0.764        0.0315
3    32    0.748     0.0177       0.730       0.765        0.0354
4     4    0.747     0.0170       0.730       0.764        0.0339
5     8    0.747     0.0170       0.730       0.764        0.0339
6    16    0.747     0.0170       0.730       0.764        0.0339
7   128    0.746     0.0186       0.728       0.765        0.0371
```

When comparing the model performances, all models performed similarly on the training data when considering the one standard deviation rule. While the XGBoost model had the highest accuracy, 77.0%, the one standard deviation rule showed this performance to be no better than the GBM and Random Forest models.

Model	Accuracy	Accuracy SD	Lower Bound	Upper Bound
GBM	76.7%	0.9%	75.8%	77.6%
Random Forest	76.4%	0.4%	76.0%	76.8%
XGBoost	77.0%	0.6%	76.3%	77.7%
SVM	74.9%	1.5%	73.4%	76.4%

When faced with these models having comparable performance, additional factors should be taken into account for the final model selection. Considerations such as interpretability and computational efficiency should play a role in selecting a model.

Model Evaluation

In this scenario, any of the trained models can be used for predictions of income. However, to understand more about the data and how income predictions vary for different individuals, the random forest was selected based on its interpretability. When assessing the model's performance on the holdout data, the random forest model predicts income level with a 76.5% accuracy. This happens to be very similar to the training accuracy of this model indicating it was not overfit nor underfit.

Figure 2

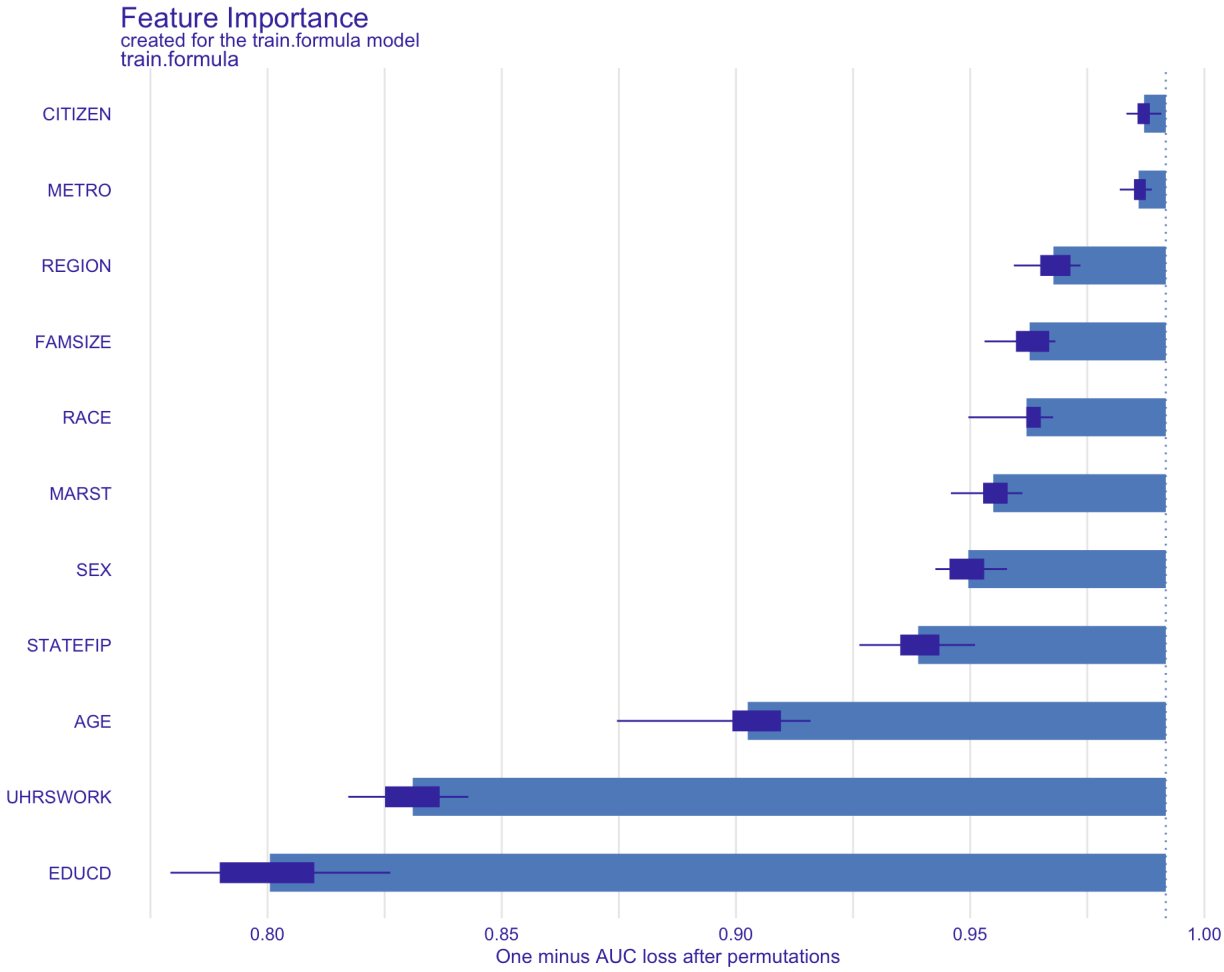


Figure 2 shows the feature importance of each variable in the Random Forest model. This assesses how much each feature contributes to the model's predictive accuracy. Here, education level (EDUCD) and usual number of hours worked per week (UHRSWORK) are the two most important variables in the model. The demographic and geographic information was shown to be less important but do still have an impact.

Figure 3

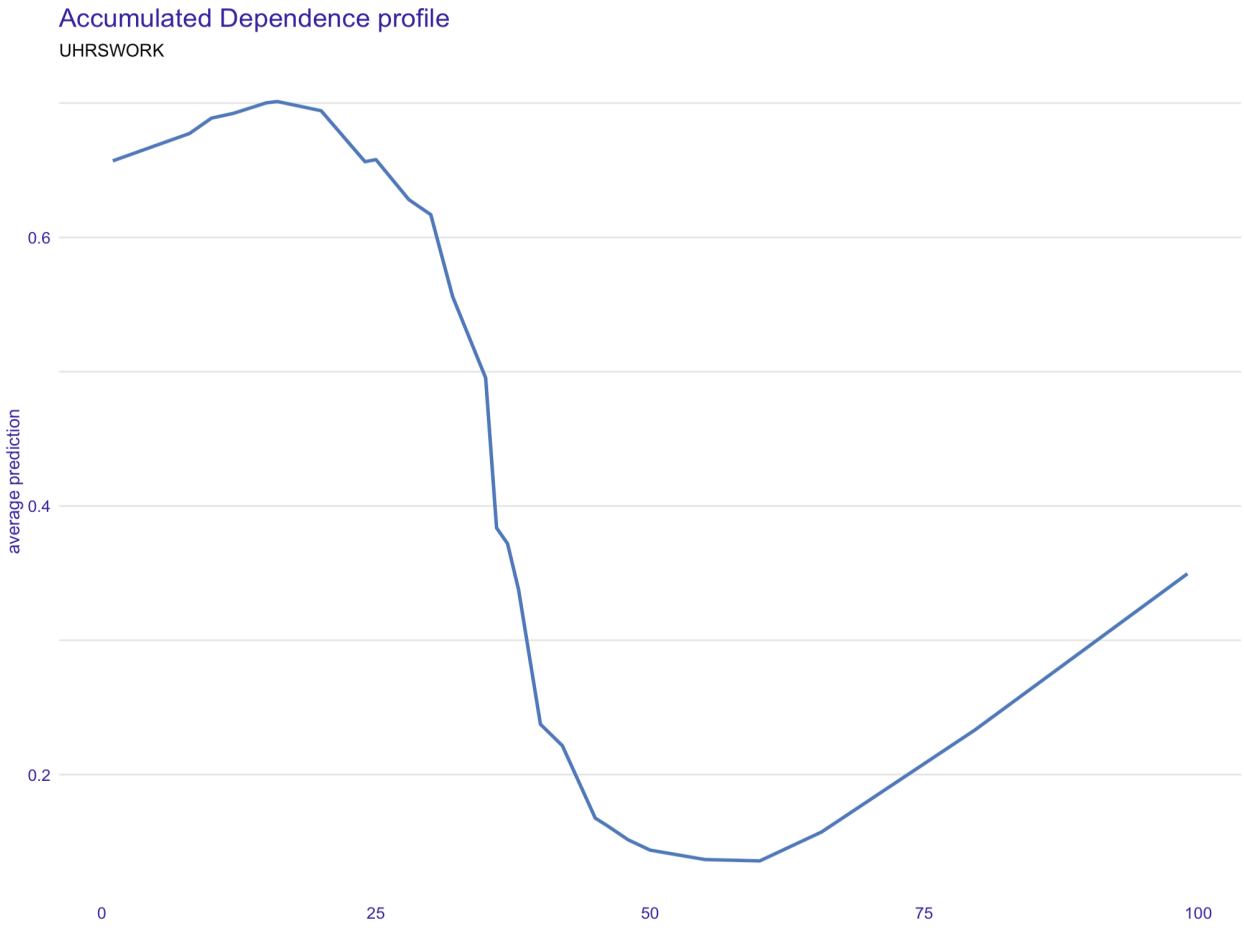


Figure 3 shows how the model's prediction changes based on the usual number of hours an individual works each week. This variable was highly important in determining the outcome of each individual and here we can see how the model's prediction changes based on this variable alone. The average prediction for being below the median income is higher for individuals who work less. The probability of being below the median income is lowest for individuals working around 50 hours per week.

Figure 4

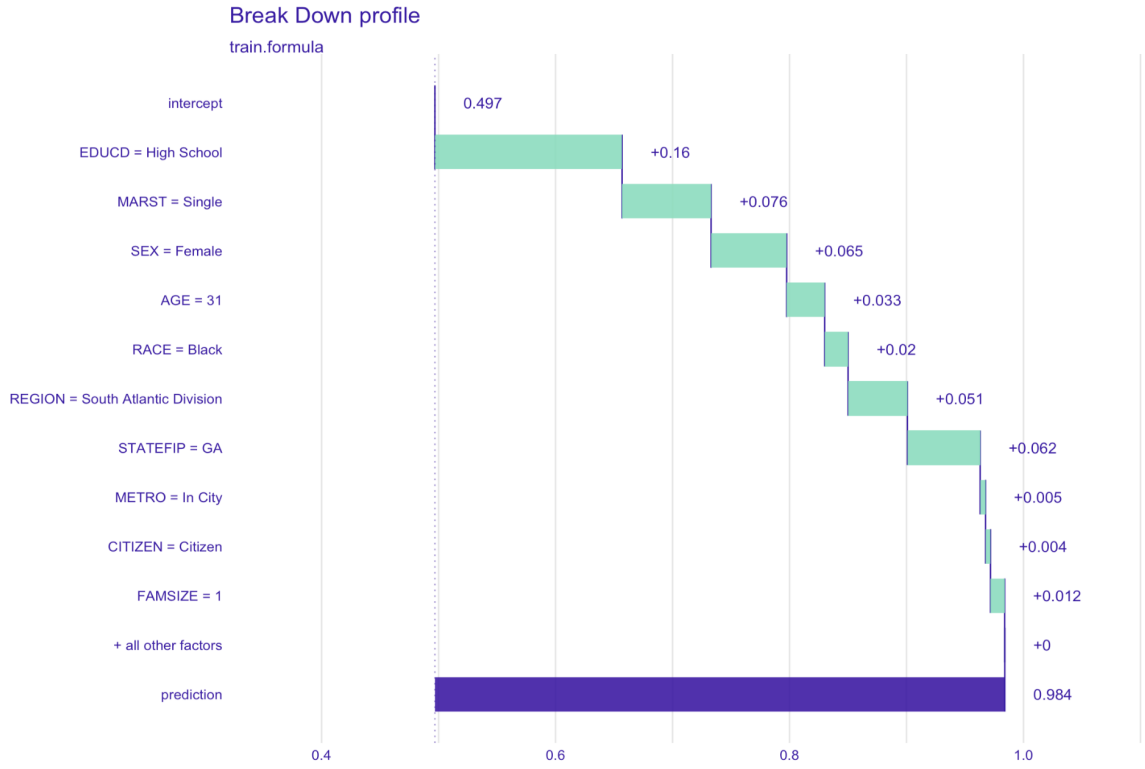


Figure 4 shows an individual who has a high probability of being below the median income. The model makes its decision by starting at 49.7% which is the model's average estimate of being above the median income. From there, it sees this individual has a high school degree which raises the probability by +16%. This individual is also single, which raises the probability by +7.6%. Combining this with the rest of the variables, this individual only has a 98.4% chance of being below the median income.

Figure 5

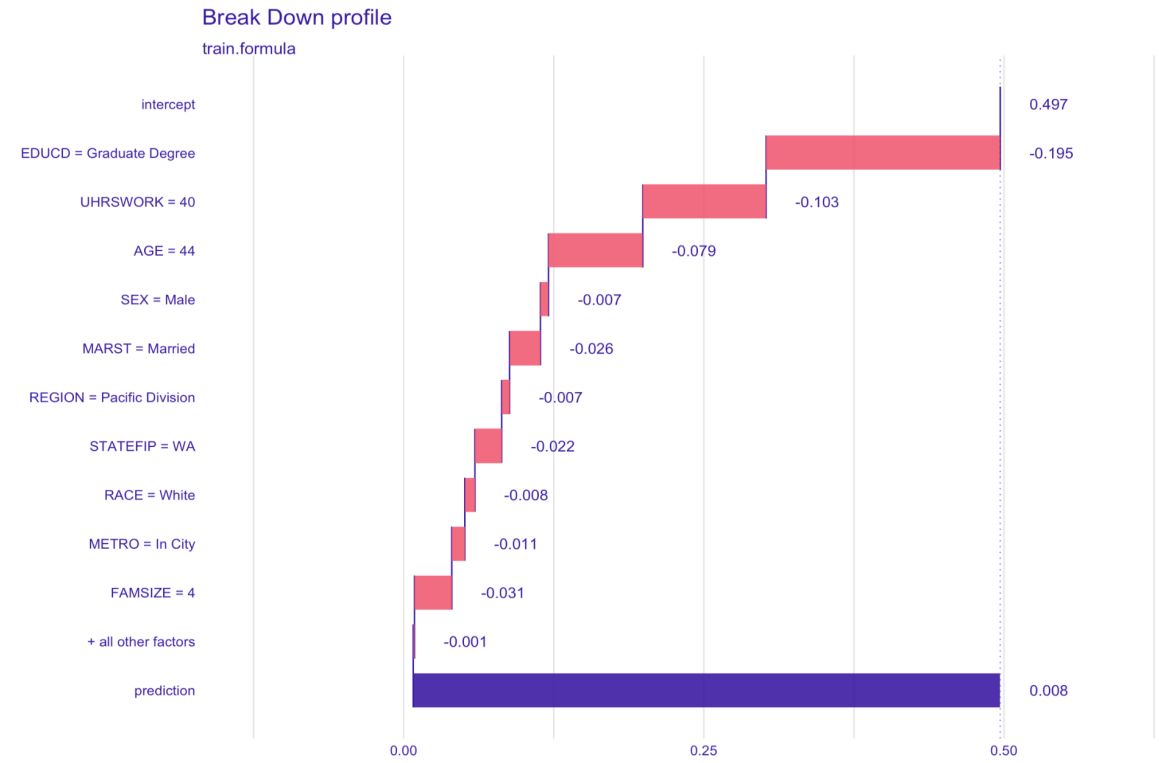


Figure 5 shows an individual who has a low probability of being below the median income. The model makes its decision by starting at 49.7% which is the model's average estimate of being below the median income. From there, it sees this individual has a graduate degree which lowers the probability by -19.5%. This individual also works only a usual 40 hours per week, which lowers their probability of being below the median income by another -10.3%. Combining this with the rest of the variables, this individual only has a 0.8% chance of being below the median income.

Conclusion

Four machine learning models - GBM, Random Forest, XGBoost, and SVM - were employed to predict whether an individual's income is above or below the median income ($45,000). While the models' accuracies ranged from 74.9% to 77.0%, the one standard deviation rule proved the models perform similarly.

The Random Forest model was chosen for further evaluation due to its interpretability. On the holdout data, the model demonstrated a consistent accuracy of 76.5%, indicating neither overfitting nor underfitting. Feature importance analysis revealed that education level (EDUCD) and usual hours worked per week (UHRSWORK) were the

most influential variables in predicting income. Examining the model's predictions, it was evident that education and work hours played crucial roles. Individuals with higher education levels and longer work hours tended to have a lower probability of falling below the median income.

Within this analysis, there were limitations of a relatively small dataset which resulted in difficulties predicting income. Future improvements may involve tuning model parameters, exploring additional features, and adapting the model for specific industry needs. Predicting income is challenging due to the complex nature of various factors that can influence income between similar individuals. This analysis was a stepping point that uncovered highly influential variables, but should be expanded upon for better performance and understanding of how to predict income in the future.

Resources

Steven Ruggles, Sarah Flood, Matthew Sobek, Daniel Backman, Annie Chen, Grace Cooper, Stephanie Richards, Renae Rogers, and Megan Schouweiler. IPUMS USA: Version 14.0 [dataset]. Minneapolis, MN: IPUMS, 2023. https://doi.org/10.18128/D010.V14.0