Predicting Income Class from Census Data

Isaiah Chastain, Derek Nelson, Lucas Dowlen, Ryan Naleway, and Dylan Smith

1. Problem Statement and Background

In this project, we will be analyzing census data from the 1994 United States Census found online in the UCI Machine Learning Repository. The purpose of this analysis is to determine if a person's income class may be predicted based off other factors shown in the dataset. The models built from our analysis will be tested on pre-partitioned test data and evaluated by their accuracy, AUC, and F1 scores (more on this in the "Results" section). This analysis is important to banks, Congress, or the IRS. Banks could use this model to predict if somebody is a good candidate for a credit line. Congress could use this model to consider changes to the tax brackets based on predicted population income class percentages. The IRS could use this model to help conduct audits. If we find that the income class can be reliably predicted, we will build a shiny application to predict this outcome for new data input.

Authors: Isaiah Chastain (25%), Dylan Smith (25%), Derek Nelson (25%), Lucas Dowlen (25%)

2. Data and Exploratory Analysis

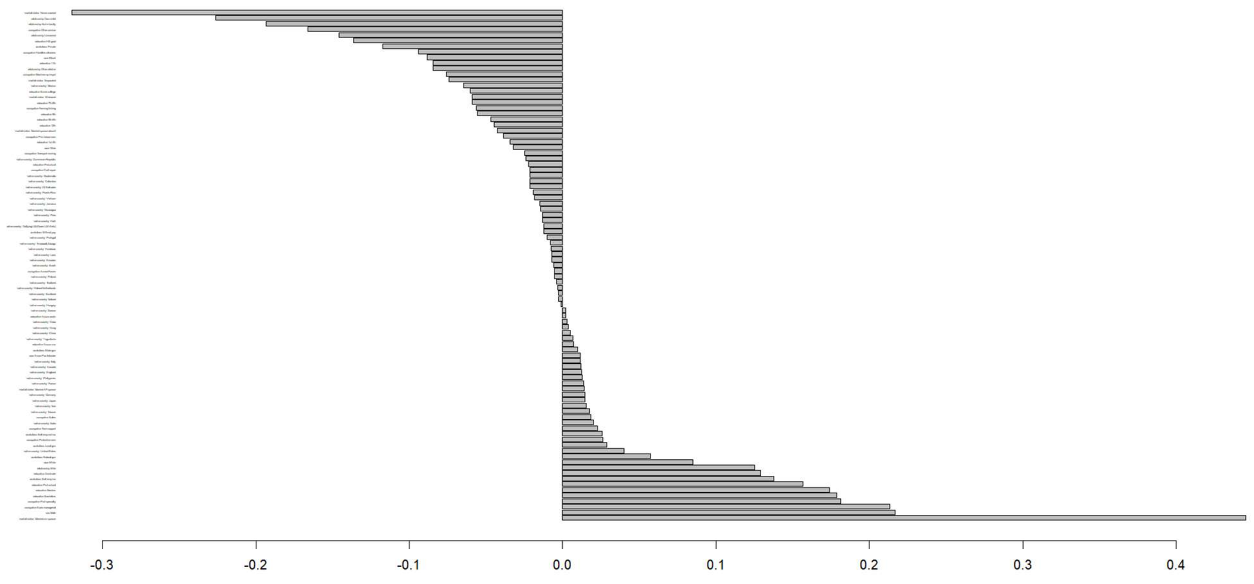
The dataset we have been given has the following columns:

Column Name	Description
age	Age of the people described in this row
workclass	Their employment status/employer
fnlwgt	The sampling weight – how many people are represented by the other statistics described in this row
education	Education level of the people described in this row
education-num	A numerical representation of this row's education level
marital-status	Marital status and if their spouse is in the military (Armed Forces)
occupation	What they do for work
relationship	What are they in their family (in relation to the head of the household)
race	What race are they
sex	What is their sex
capital-gain	Monetary gain through sale of assets
capital-loss	Monetary loss through sale of assets
hours-per-week	How many hours worked in a week

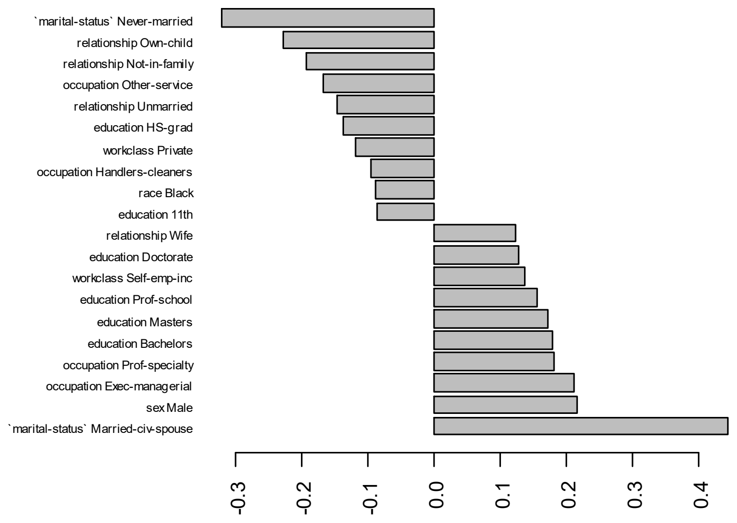
native-country	What country are they from
class	Whether the people described by this row make less than or equal to $50K annually, or more than $50K annually

One important thing to note is that in the original dataset, the NA values were *not* filled with NA, they were filled with " ?". Our first step was to convert these question marks into NA's using lapply and an ifelse branch. Secondly, we realized that all of our NA values were in columns that contained categorical values, not numerical. For this reason, and the fact that our dataset is very large, we chose to simply drop the rows containing NA values instead of imputing data into them. This left us with 30,161 of our original 32,560 entries. The only data that we generated was our class column - instead of having the char datatype categories "<= $50K" and ">$50K", we converted these values to FALSE and TRUE logical values respectively. This dataset has ***many*** outliers, however, for the most part these outliers are not errors – they are valid entries in which the people genuinely just made far more than the general population. We cannot get rid of these entries, as it would skew the data incorrectly. This dataset does seem to max out at $99,999 for values that may possibly be greater than $99,999 in the "capital-gain" column. After cleaning our data, we created correlation coefficients to see which factors tended to affect the probability an instance being "class = TRUE" (the person represented by the data makes more than $50K a year). We found that the three most impactful coefficient correlations were: marital-status = Married-civ-spouse (+0.445), marital-status = Never-married (-0.32), and relationship = Own-child (-0.226).

Here is the full correlation graph:



As these legends are too small to read, here is a graph of the 10 most positive and 10 most negative correlations:



Authors: Isaiah Chastain (25%), Dylan Smith (25%), Lucas Dowlen (25%), Ryan Naleway (25%)

3. Methods

Our initial step in tackling this problem is data cleaning – as mentioned before, we had to convert the " ?" entries to NA's and then drop the rows containing those NA's. After cleaning our data, we converted our "class" column from a char datatype to a logical datatype. We also created correlation coefficients that determined the weight and direction of a variable's influence on whether the person described by that variable would make over $50K. As our machine learning model needs to do a classification task, we considered using a logistic model, a random forest model, a naïve bayes model, or an XGBoost model. All of these models did work, however, by evaluating accuracy, precision, and AUC, we discovered that some were more proficient than others at predicting the income class. Individual model performance will be discussed in the Results section. At this point, we realized that we could make an application, as the models could predict the income class with relatively high accuracy.

Authors: Isaiah Chastain (20%), Derek Nelson (20%), Dylan Smith (20%), Lucas Dowlen (20%), Ryan Naleway (20%)

4. Tools

For our project, since this is a classification issue, we must use a machine learning model. We used the following libraries: ggplot2 for visualization; tidymodels, dplyr, randomForest, xgboost, discrim, naivebayes, patchwork, vip, pdp, caret, and cvms for machine learning model integration and experimentation; and shiny for application development. The general process we used for integrating a model is as such:

1) Split training and testing data with initial_split()
2) Define the model with its parameters
3) Define a recipe (recipe <- recipe(class ~ ., adults) for all models)
4) Define a workflow and add the recipe and model (workflow() |> add_recipe() |> add_model)
5) Fit the model to our training data
6) Predict the outcome of the model on our testing data

Once we predicted the outcomes, we bound the columns of our testing data to our predicted outcome data frame for easier metric calculations (predict(fitted_model, new_data = test) |> bind_cols(test)). From this point, we began testing our individual models to see which one performed the best. Once we determined which model did perform the best, we saved it so that we could integrate it later into our app. Our method to integrate this model was as such:

- Receive new input from the user of the application using input buttons/fields in the shiny package.
- Create a dummy data frame containing only the new information input.
- Use the pre-trained model to predict income class based off the user input.
- Output the result using uiOutput from the shiny package.

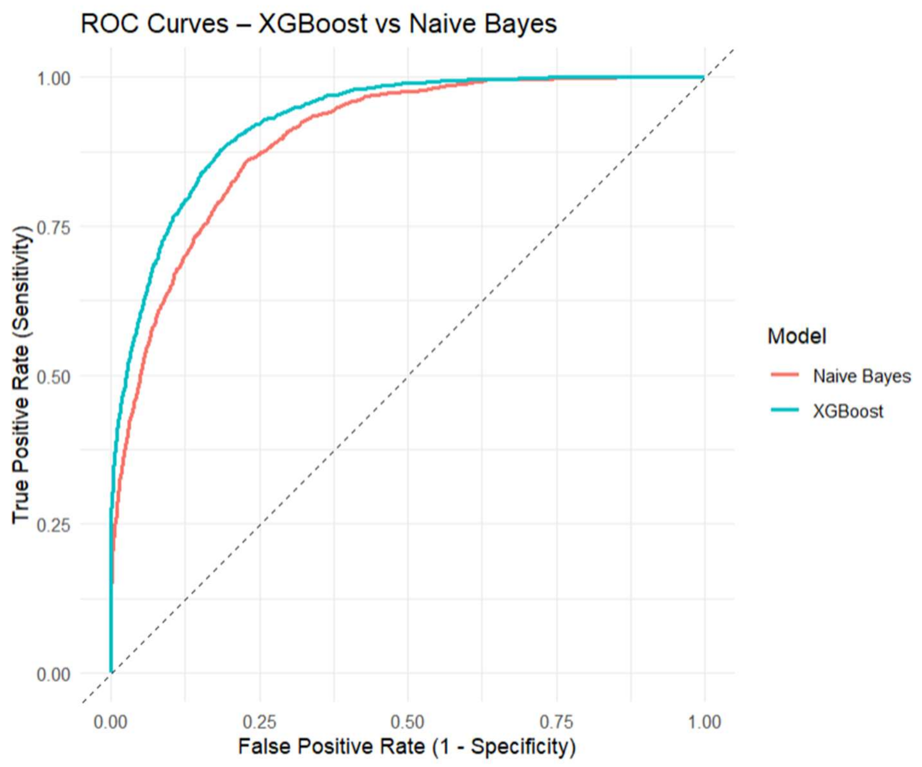
Authors: Isaiah Chastain (20%), Derek Nelson (20%), Dylan Smith (20%), Lucas Dowlen (20%), Ryan Naleway (20%)

5. Results

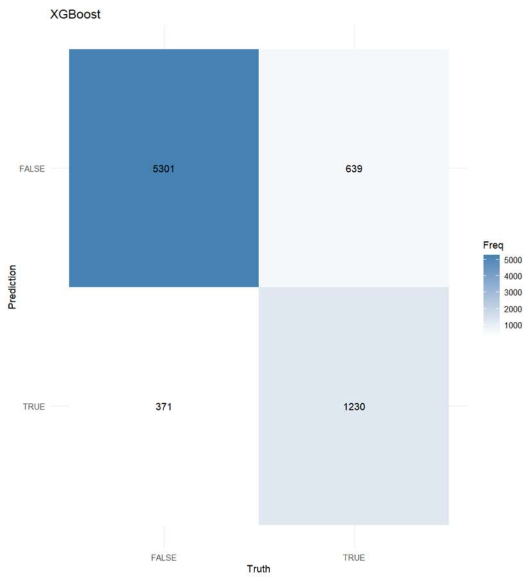
We will begin by providing a table of different model performance metrics to determine which model we should use (red = worst performance, yellow = in-between performance, green = best performance):

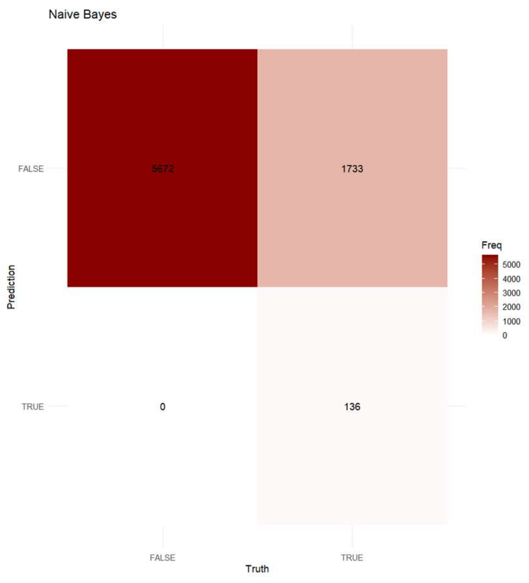
Model Name	**Accuracy**	**F1**	**ROC AUC**
Naïve Bayes	0.770	0.867	0.897
Logistic Classification	0.847	0.901	0.907
Random Forests	0.862	0.911	0.901
XGBoost	0.866	0.913	0.927

This table brings us to our conclusion – our XGBoost model outperformed every other model in terms of accuracy, F1 score, and AUC. It also ran second fastest, beat only by the naïve Bayes model. For deeper analysis of these models, we decided to make visualizations comparing our best model (XGBoost) compared to our worst model (Naïve Bayes). Here is the graph of these models' ROC curves:



And their confusion matrices:





This is why, as we continued to our end goal of developing an application to predict the income class of an individual, we decided to integrate the XGBoost model into our shiny application. In order to do this, we first saved the individual XGBoost model and implemented it into the back end of our app using the description of this process in **Tools**. At this point, we cleaned up our front-end and had a working application that predicts income class with 86.6% accuracy!

Authors: Isaiah Chastain (20%), Derek Nelson (20%), Dylan Smith (20%), Lucas Dowlen (20%), Ryan Naleway (20%)

6. Summary and Conclusions

In this project, we decided to analyze a dataset containing the census data from the United States 1994 census. Our end goal was to a) determine if predicting income class from existing census data was possible, and b) if it was, to make an application in which you could enter data, and be told whether the person with the data you entered will make less than or equal to $50K, or more than $50K (at least, in 1994). We discovered that, not only was it possible to predict the person's income class, but it was also possible to predict it with **86.6% accuracy**. We have now developed an app available in the "CensusApplication" folder in the repository below that implements the model we trained and tells you clearly what income class your input person will be in!

Authors: Isaiah Chastain (100%)

7. Appendix

Dataset: https://archive.ics.uci.edu/dataset/2/adult

Github Repo: https://github.com/IsaiahStain05/CSC-3220-Project.git

Documentation:

- Ggplot2: https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf
- Tidymodels: https://cran.r-project.org/web/packages/tidymodels/tidymodels.pdf
- Dplyr: https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
- randomForest: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
- xgboost: https://cran.r-project.org/web/packages/xgboost/xgboost.pdf
- discrim: https://cran.r-project.org/web/packages/discrim/discrim.pdf
- naivebayes: https://cran.r-project.org/web/packages/naivebayes/naivebayes.pdf
- caret: https://cran.r-project.org/web/packages/caret/caret.pdf
- cvms: https://cran.r-project.org/web/packages/cvms/cvms.pdf

- pdp: https://cran.r-project.org/web/packages/pdp/pdp.pdf
- vip: https://cran.r-project.org/web/packages/vip/vip.pdf
- tidyverse: https://cran.r-project.org/web/packages/tidyverse/tidyverse.pdf
- shiny: https://cran.r-project.org/web/packages/shiny/shiny.pdf

Authors: Isaiah Chastain (100%)